Bài thi thử (Buổi chiều)
Kỹ sư IT cơ bản

Các câu hỏi phải được trả lời theo quy định dưới đây:

Câu hỏi số	Q1 - Q5	Q6 - Q7	Q8 - Q9
Lựa chọn câu hỏi	Bắt buộc	Lựa chọn 1 trong 2	Lựa chọn một trong 2
Thời gian thi	150 phút		

[Giải thích các định dạng giả mã lệnh]

Cú pháp ngôn ngữ giả mã lệnh	Diễn tả
	Vùng liên tục nơi mô tả khai báo và tiến trình
∘	Tên khai báo, kiểu… của thủ tục, biến…
▪ biến ← Biểu thức	Gán giá trị của một biểu thức cho một biến
Biểu thức điều kiện ▪ Công việc 1 [▪ Công việc 2]	Tiến trình lựa chọn. Nếu biểu thức điều kiện là đúng, thì công việc 1 được thực hiện [tuỳ chọn] Nếu điều kiện là sai, thì công việc 2 được thực hiện.
Biểu diễn điều kiện ▪ Công việc	Một tiến trình lặp đi lặp lại với điều kiện ở đầu tiên. Công việc còn được thực hiện trong khi biểu thức điều kiện còn đúng.

[Toán tử]

Phép toán	Toán tử	Mức độ ưu
Phép toán một toán hang	+ - not	Cao Thấp
Phép toán nhân và chia	* /	
Phép toán cộng và trừ	+ -	
Phép toán quan hệ	> < >= <= = ≠	
Nhân logic	and	
Cộng logic Phép toán cộng logic	Or Xor	

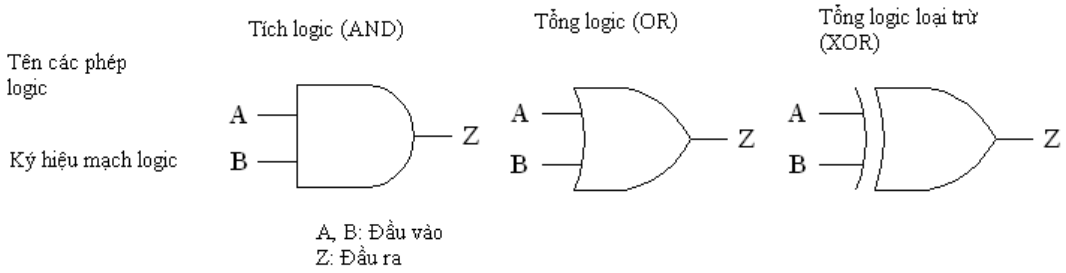
[hằng số kiểu logic]

```
true  false
```

Từ câu 1 đến câu 5 là bắt buộc. Hãy trả lời tất cả các câu.

Q1. Đọc các diễn tả sau về các phép toán logic và bộ cộng đầy đủ, sau đó trả lời các câu hỏi con từ 1 đến 3.

(1) Các ký hiệu mạch logic cho các phép toán logic chính như sau.



(2) Hình dưới đây mô tả bộ cộng đầy đủ, cộng các số nhị phân có nhớ. Bảng biểu diễn bảng thật của bộ cộng đầy đủ.

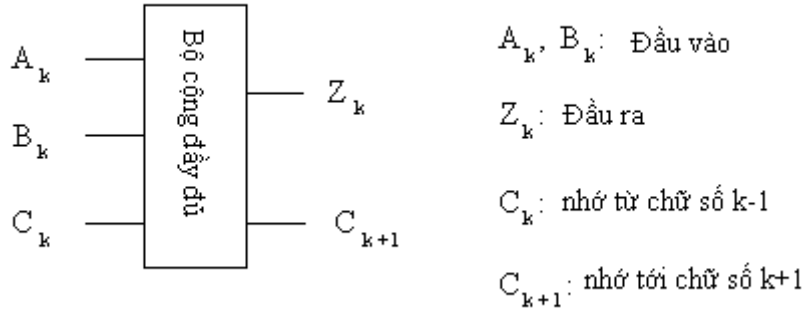


Fig. Bộ cộng đầy đủ (k chữ số)

Bảng Bảng thật của bộ cộng đầy đủ

Input			Output	
C_k	A_k	B_k	C_{k+1}	Z_k
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1		
1	1	0	1	0

1	1	1	1	1

Câu hỏi con 1

Từ nhóm các câu trả lời sau, lựa chọn câu trả lời chính xác để điền vào chỗ trống ☐ trong bảng thật của bộ cộng đầy đủ.

Nhóm câu trả lời:

a)

0	0
1	0

b)

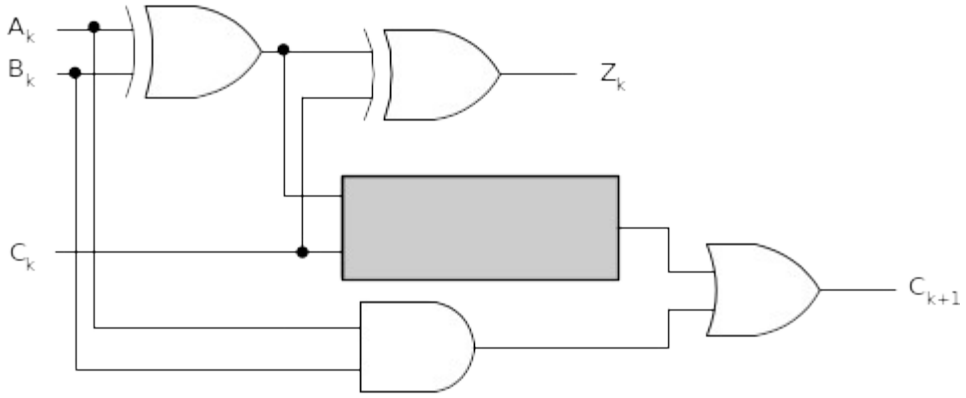
0	1
1	1

c)

d)

Câu hỏi con 2

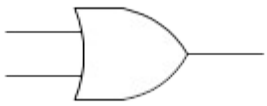
Từ nhóm các câu trả lời sau, lựa chọn câu trả lời chính xác để điền vào chỗ trống ▭ trong mạch logic của bộ cộng đầy đủ.



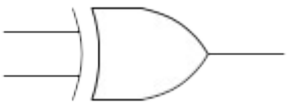
Nhóm câu trả lời

a)

b)

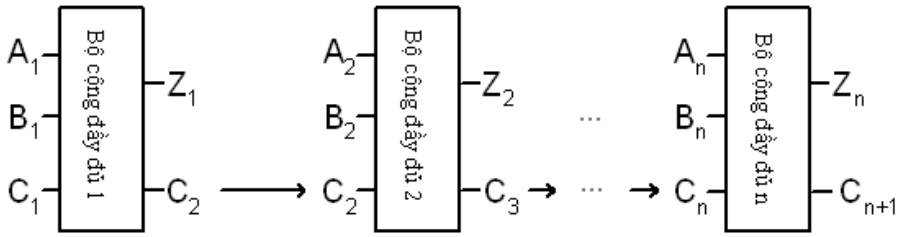


c)

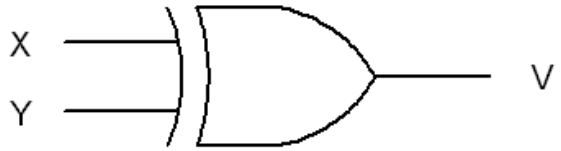


Câu hỏi 3

Khi một mạch logic được cấu hình với bộ cộng đầy đủ để cộng số nhị phân n chữ số biểu diễn dưới dạng số bù hai, phép cộng của các chữ số có ý nghĩa nhất (A_n, B_n và C_n) là nguyên nhân gây ra tràn (phần tô đậm trong bảng thật của bộ cộng đầy đủ). Một mạch logic cho việc phát hiện điều này bằng cách cấu hình với một phần tử XOR. Chọn từ nhóm các câu trả lời sau đây sự kết hợp đúng của các đầu vào X và Y cho mạch logic này.



Chú ý: C1 được gán = 0



Tràn khi V = 1

Nhóm câu trả lời:

a) A_n, B_n
b) A_n, Z_n
c) B_n, Z_n
d) C_n, C_{n+1}
e) C_n, Z_n
f) C_{n+1}, Z_n

Q2. Đọc các mô tả dưới đây về cơ sở dữ liệu quan hệ. Sau đó trả lời các câu hỏi con từ 1 đến 3.

Cơ sở dữ liệu quan hệ dưới đây gồm có một bảng Nhân viên và một bản Kỹ năng nhân viên.

Bang_Nhan_Vien

MaNhanVien	TenNhanVien	Phong
0001	Brown	A1
0002	Charles	A2
0003	Taylor	B1
0004	Williams	D3
0005	Parker	A1
0006	James	B1

Bang_Ky_Nang_Nhan_Vien

MaNhanVien	MaKyNang	NgayDangKy
0001	FE	19991201
0001	DB	20010701
0002	NW	19980701
0002	FE	19990701
0002	SW	20000701
0005	NW	19991201

Câu hỏi con 1.

Từ các nhóm câu trả lời phía dưới, chọn các câu trả lời đúng để điền vào các ô trống [ ] trong đoạn mô tả sau.

True được trả về cho mệnh đề EXISTS khi kết quả của câu truy vấn con tồn tại và False được trả về nếu không tồn tại. Khi câu truy vấn SQL sau được thực thi thì số lượng nhân viên được chọn là [A].

```
SELECT MaNhanVien FROM Bang_Ky_Nang_Nhan_Vien
            WHERE EXISTS (SELECT * FROM Bang_Ky_Nang_Nhan_Vien
                        WHERE MaKyNang = 'FE')
```

Nếu câu truy vấn con chứa tham chiếu đến một bảng khác trong câu truy vấn chính thì Mệnh đề EXISTS đánh giá từng hàng tương ứng với tham chiếu đó. Khi câu truy vấn SQL sau được thực thi thì số lượng nhân viên được chọn là [B].

```
SELECT TenNhanVien FROM BangNhanVien A
            WHERE EXISTS (SELECT * FROM Bang_Ky_Nang_Nhan_Vien B
                        WHERE MaKyNang = 'FE'
                        AND A.MaNhanVien = B.MaNhanVien)
```

Nhóm câu trả lời cho A và B:

a) 0 b) 1 c) 2 d) 3
e) 4 f) 5 g) 6

Câu hỏi con 2.

Câu truy vấn SQL sau đưa ra một số mã nhân viên. Nhân viên nào sẽ được chọn?
Từ nhóm câu trả lời dưới đây, chọn câu trả lời đúng.

```
SELECT DISTINCT MaNhanVien FROM Bang_Ky_Nang_Nhan_Vien B1
          WHERE EXISTS (SELECT * FROM Bang_Ky_Nang_Nhan_Vien B2
                    WHERE B1.MaNhanVien = B2. MaNhanVien
                    AND B1.MaKyNang < > B2.MaKyNang)
```

Nhóm câu trả lời:

a) Những nhân viên nào có ít nhất một kỹ năng.

b) Những nhân viên nào chỉ có duy nhất một kỹ năng.
c) Những nhân viên nào không có kỹ năng.
d) Những nhân viên nào có nhiều kỹ năng.

Câu hỏi con 3.

Bạn muốn thêm tên nhân viên vào thông tin thu được từ Câu hỏi 2. Từ nhóm câu trả lời dưới đây, chọn câu trả lời đúng để điền vào ô trống ☐ trong câu truy vấn SQL sau.

```
SELECT DISTINCT A.MaNhanVien, TenNhanVien
                    FROM Bang_Nhan_Vien A, Bang_Ky_Nang_Nhan_Vien B1
          WHERE EXISTS (SELECT * FROM Bang_Ky_Nang_Nhan_Vien B2
                    WHERE B1. MaNhanVien = B2. MaNhanVien
                    AND B1.MaKyNang < > B2. MaKyNang)
                    [                                  ]
```

Nhóm câu trả lời:

a) AND A.MaNhanVien = B1. MaNhanVien
b) BETWEEN A. MaNhanVien AND B1. MaNhanVien
c) LIKE A. MaNhanVien = B1. MaNhanVien
d) OR A. MaNhanVien = B1. MaNhanVien

Q3. Đọc đoạn mô tả chương trình sau, sau đó trả lời các câu hỏi từ 1 đến 3.

[Mô tả chương trình]

Chương trình này thực thi chức năng của một trình soạn thảo văn bản.

(1) Trình soạn thảo văn bản này có thể chứa 80 ký tự trên một dòng và tối đa lên đến MAX dòng.

(2) Biến toàn cục sau được khai báo

(i) `CP`: Biến này chỉ ra dòng được xử lý

(3) Các chương trình con sau được định nghĩa

(i) `INSERT(x)`: Thêm một dòng mới x tại dòng được chỉ bởi CP. Việc thêm này sẽ làm cho dòng hiện tại và các dòng sau đó bị đẩy xuống 1 dòng.

(ii) `DELETE()`: Xóa dòng chỉ bởi CP. Việc xóa này làm cho dòng tiếp theo và các dòng sau đó đẩy lên 1 dòng.

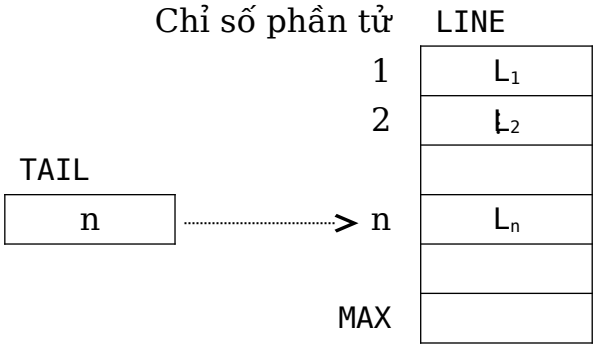
(iii) `LAST()`: Trả về thứ tự dòng dòng cuối (số dòng của văn bản). Nếu văn bản rỗng trả về "0".

(iv) `GET()`: Trả về xâu ký tự trên dòng chỉ bởi CP.

Câu hỏi con 1.

Một xâu ký tự kiểu mảng `LINE[i]` (`i` = 1, 2, ..., `MAX`) và biến kiểu nguyên TAIL được định nghĩa trong chương trình. Xâu ký tự L_i trên dòng i được lưu trữ trong `LINE[i]`và một chỉ số của phần tử mảng ("0" nếu rỗng) luôn được gắn với dòng cuối cùng được lưu trữ trong biến TAIL.

Biến CP chứa một chỉ số của phần tử mảng ứng với dòng được xử lý. Từ nhóm câu trả lời dưới đây, chọn chương trình con mà tổng số lượng (theo trình tự) tính toán cần thiết là một hằng số không phụ thuộc số dòng văn bản (chọn **<u>tất cả</u>** các câu trả lời thích hợp).



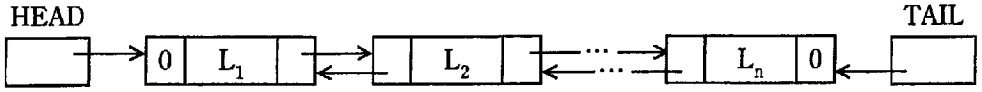
Hình 1. Ví dụ: Sử dụng mảng

Nhóm câu trả lời:

a) `DELETE ( )` b) `GET ( )` c) `INSERT (x)` d) `LAST ( )`

Câu hỏi con 2.

Chương trình đã được thay đổi để sử dụng một danh sách móc nối kép dùng con trỏ như hình 2. Phần tử i trong danh sách bao gồm con trỏ trỏ đến phần tử chứa dòng `i−1`, xâu ký tự `Li` ở dòng i và con trỏ trỏ đến phần tử chứa dòng i+1. Nếu một phần tử nào đó không tồn tại giá trị con trỏ sẽ là "0".
Biến HEAD là một con trỏ trỏ đến phần tử chứa dòng đầu tiên. Biến TAIL là con trỏ trỏ đến phần tử chứa dòng cuối cùng.
Biến CP là con trỏ trỏ đến phần tử chứa dòng đang được xử lý. Tử nhóm câu trả lời dưới đây chọn chương trình con mà tổng số lượng (theo trình tự) tính toán cần thiết là một hằng số không phụ thuộc vào số dòng văn bản (chọn **tất cả** các câu trả lời thích hợp).



Hình 2. Ví dụ: Sử dụng con trỏ

Nhóm câu trả lời:

a) `DELETE ( )` b) `GET ( )` c) `INSERT (x)` d) `LAST ( )`

Câu hỏi con 3.

Từ nhóm câu trả lời dưới đây, chọn các câu trả lời đúng điền vào các ô trống ☐ trong đoạn sau:

Một danh sách liên kết 2 chiều sử dụng con trỏ được thực thi bằng cách dùng 3 mảng. Hình 3 là một ví dụ của trường hợp mà số dòng tối đa là 10. Biến CP chứa chỉ số của các phần tử mảng chứa dòng được xử lý. Biến HEAD chứa chỉ số phần tử mảng chứa dòng đầu tiên. Biến TAIL chứa chỉ số của phần tử mảng chứa dòng cuối cùng. Biến EMPTY chứa chỉ số của phần tử mảng đầu tiên trong danh sách rỗng chứa.
Giả sử rằng các dòng L_1, L_2, L_3, L_4, và L_5 được lưu trữ như hình 3 và CP=8. Nếu `DELETE()` được thực thi để xóa phần tử mảng chứa L_3 sau đó giá trị của `HEAD` và `TAIL` không thay đổi và `EMPTY` = 8, `PREV[9]` = [A], `NEXT[2]` = [B], và `NEXT[8]` = [C]. Giả sử rằng phần tử mảng bị xóa được thêm vào đầu danh sách rỗng.

CP	Số liệu phần tử	PREV	LINE	NEXT
8	1			0
	2	4	L_2	8
HEAD	3			5
4	4	0	L_1	2
	5			7
TAIL	6	9	L_5	0
6	7			10
	8	2	L_3	9
EMPTY	9	8	L_4	6
3	10			1

Hình 3. Ví dụ: Thực thi danh sách liên kết 2 chiều sử dụng mảng

Nhóm câu trả lời:

a) 0 b) 1 c) 2 d) 3
e) 7 f) 8 g) 9 h) 10

Q4. Đọc đoạn mô tả và chương trình dưới đây rồi trả lời các Câu hỏi con 1, 2, và 3.

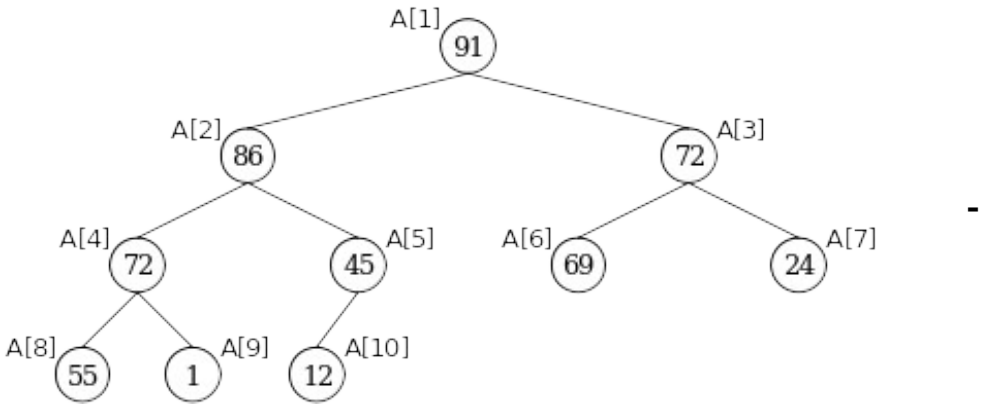
[Mô tả chương trình]

Chương trình con `HeapSort` là một chương trình sắp xếp các số nguyên được lưu trữ trong một mảng theo thứ tự tăng dần bởi phương pháp vun đống.

(1) `Num` số nguyên (`Num >= 2`) cần sắp xếp được lưu trữ trong một mảng các biến toàn cục `A[1]`, `A[2]`, ... , `A[Num]`.

(2) Phương pháp vun đống dùng cây nhị phân để sắp xếp dữ liệu. Biểu diễn một cây nhị phân lưu trữ các phần tử của mảng theo cách sau: nếu một nút nào đó lưu trữ giá trị của phần tử `A[i]` thì nút con bên trái của nó lưu trữ giá trị của phần tử `A[2*i]` và nút con bên phải lưu trữ giá trị của phần tử `A[2*i+1]`. Trong hình vẽ dưới đây, các ô tròn biểu diễn các nút, số bên trong ô tròn là giá trị của nút, và bên cạnh các ô tròn là các phần tử tương ứng của mảng.

(3) Như trong hình vẽ dưới đây, đống là một cây nhị phân trong đó giá trị của mỗi nút lớn hơn hoặc bằng giá trị tại các nút con của nó.



Ví dụ về một đống

(4) Thủ tục sắp xếp như sau.

(i) Các phần tử `A[1]`, `A[2]`, ... , `A[Num]` của mảng `A` là các phần tử cần sắp xếp.

(ii) Các phần tử cần sắp xếp biểu diễn qua một cây nhị phân như theo mô tả ở (2), và đống được tạo ra bằng cách đổi chỗ giá trị các phần tử. Do đó, tại gốc của cây (`A[1]`) lưu trữ giá trị lớn nhất trong số các phần tử cần sắp xếp.

(iii) Đổi chỗ giá trị tại nút gốc của cây (`A[1]`) và giá trị tại nút cuối cùng của cây (là nút tương đương với phần tử cuối cùng cần sắp xếp).

(iv) Nút cuối cùng của cây được gỡ khỏi cây nhị phân (Số các phần tử được lưu trữ giảm đi 1.)

(v) Quá trình từ Bước (ii) đến bước (iv) được lặp lại cho đến khi chỉ còn gốc của cây.

(5) Thủ tục xây dựng đống như sau:

(i) Tạo gốc của cây là nút cha.

(ii) Kết thúc nếu không có nút con.

(iii) So sánh giá trị của nút con (là nút có giá trị lớn hơn trong hai nút con) với giá trị của nút cha, và đổi chỗ giá trị của chúng nếu giá trị của nút cha nhỏ hơn giá trị của nút con, ngược lại chương trình sẽ dừng nếu giá trị của nút cha lớn hơn hoặc bằng giá trị của nút con.

(iv) Lặp lại Bước (i) đến bước (iii) cho cây con có gốc là nút có giá trị đã được đổi chỗ.

(6) Đặc tả các tham số của chương trình con được thể hiện trong các Bảng từ 1 đến 4.

Bảng 1 Đặc tả các Tham sô´của HeapSort

Tên Tham số	Dạng Dữ liệu	Đầu vào / Đầu ra	Ý nghĩa
Num	Số nguyên	Đầu vào	Chỉ số phần tử mảng tương ứng với nút cuối cùng của cây

Bảng 2 Đặc tả các Tham số của InitHeap

Tên Tham số	Dạng Dữ liệu	Đầu vào / Đầu ra	Ý nghĩa
Last	Số nguyên	Đầu vào	Chỉ số phần tử mảng tương ứng với nút cuối cùng của cây biểu diễn đống được vun lần đầu

Bảng 3 Đặc tả các Tham số của Make Heap

Tên Tham số	Dạng Dữ liệu	Đầu vào / Đầu ra	Ý nghĩa
Top	Số nguyên	Đầu vào	Chỉ số phần tử của mảng tương ứng gốc cây con biểu diễn đống được vun lại.
Last	Số nguyên	Đầu vào	Chỉ số phần tử của mảng tương ứng nút cuối cùng của cây con biểu diễn đống được vun lại.

Bảng 4 Đặc tả các Tham số của Swap

Tên Tham số	Dạng Dữ	Đầu vào /	Ý nghĩa
X	Số nguyên	Đầu vào	Chỉ số phần tử mảng để đổi chỗ với A[Y]
Y	Số nguyên	Đầu vào	Chỉ số phần tử mảng để đổi chỗ với A[X]

[Chương trình]

```
○Integer type: A[1000000]/*Sử dụng như một biến toàn cục*/

○HeapSort(Integer type: Num)
○Integer type: Idx
                                    /* Vun đống lần đầu */
▪ InitHeap(Num)
                                    /* Sắp xếp */
■ Idx: Num, Idx > 1, -1
    ▪ Swap(1, Idx)
    ▪ MakeHeap(1, Idx-1)
■


○MakeHeap(Integer type: Top, Integer type: Last)
○Integer type: L, R
▪ [ A ]
▪ R ← L + 1
▲ R <= Last                         /* So sánh 3 phần tử */
    ▲ A[L] < A[R]                   /* Phần tử phải lớn hơn */
        ▲ A[Top] < A[R]
            ▪ Swap(Top, R)
            ▪ MakeHeap(R, Last)
    ——————                          /* Phần tử trái lớn hơn */
        ▲ A[Top] < A[L]
            ▪ Swap(Top, L)
            ▪ MakeHeap(L, Last)
——————————
    [ B ]                           /* So sánh hai phần tử */
    ▲ A[Top] < A[L]
        ▪ Swap(Top, L)
        ▪ MakeHeap(L, Last)
▼
○Swap(Integer type: X, Integer type: Y)
○Integer type: Tmp
▪ Tmp ← A[X]
▪ A[X] ← A[Y]
▪ A[Y] ← Tmp
```

Câu hỏi con 1

Từ các tập câu trả lời ở dưới, lựa chọn câu trả lời chính xác chèn vào các ô trống [ ]
trong chương trình ở trên.

Tập câu trả lời cho A:

a) `L ← Top`
b) `L ← Top + 1`
c) `L ← Top * 2`
d) `L ← Top * 2 + 1`

Tập câu trả lời cho B:

a) `L <= Last`
b) `L < Last`
c) `R <= Last - 1`
d) `R <= Last - 2`

Câu hỏi con 2

Từ tập câu trả lời ở dưới, lựa chọn câu trả lời chính xác chèn vào các ô trống [ ]
trong đoạn mô tả sau đây.

Sử dụng đống trong hình vẽ, với các Bước (iii) và (iv) của (4) trong đoạn [Mô tả chương trình] được thực thi một lần và Bước (ii) được hoàn thành, thì chỉ số phần tử mảng lưu trữ giá trị 12 (ban đầu được lưu trữ trong `A[10]`) là [C]. Số lần chuyển đổi giá trị các nút ở đây là [D].

Tập câu trả lời:

a) 2
b) 3
c) 4
d) 5
e) 6
f) 7
g) 8
h) 9

Câu hỏi con 3

Chương trình con `InitHeap` vun đống lần đầu được xây dựng nhờ sử dụng chương trình con `Make Heap`. Từ tập câu trả lời ở dưới, lựa chọn câu trả lời chính xác nhất để điền vào ô trống [ ] trong chương trình sau đây.

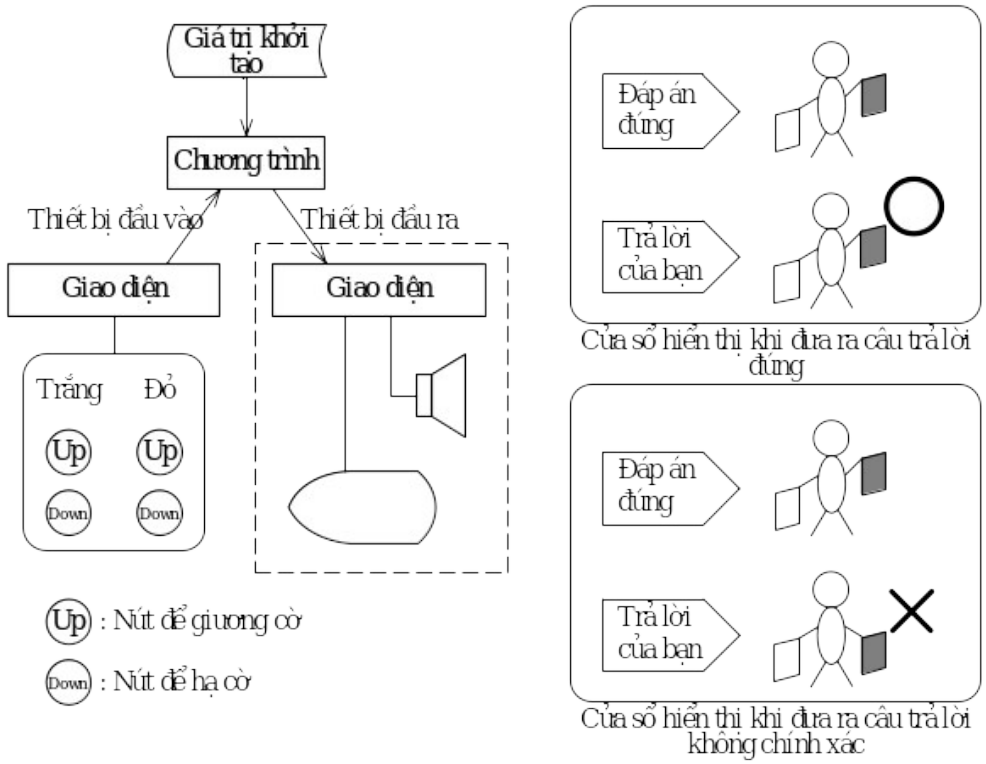
```
○ InitHeap(Integer type: Last)
○ Integer type: Idx
■ [ ]
    ▪ Make Heap(Idx, Last)
■
```

Tập câu trả lời:

a) `Idx: 1, Idx <= Last, 2`
b) `Idx: 1, Idx <= Last / 2, 1`
c) `Idx: Last, Idx >= 1, -2`
d) `Idx: Last / 2, Idx >= 1, -1`

Q5. Đọc phần mô tả dưới đây về thiết kế chương trình rồi trả lời các câu hỏi con 1 và 2.

Bạn dự định thiết kế một chương trình trò chơi giương cờ và hạ cờ. Trong trò chơi này, người chơi đáp lại các chỉ lệnh (giương cờ/hạ cờ đỏ và/hoặc cờ trắng) sử dụng thiết bị nhập bằng cách thao tác trên các nút bấm. Một loại cửa sổ sẽ hiển thị ra khi câu trả lời chính xác và một loại cửa sổ khác hiển thị khi câu trả lời không chính xác. Mô hình này chỉ được lặp lại trong một số lần xác định và số câu trả lời chính xác sẽ được hiển thị thành điểm số của người chơi. Cấu hình phần cứng và các cửa sổ của trò chơi cho các câu trả lời chính xác và không chính xác được trình bày trong Hình 1 dưới đây.



Hình 1. Cấu hình phần cứng và hiển thị trả lời đúng/sai của trò chơi.

- Thiết bị xuất sẽ đưa ra một chỉ lệnh bằng giọng nói và hiển thị câu trả lời đúng và câu trả lời của người chơi sau khi hết thời gian phát hiện câu trả lời. Khi trò chơi kết thúc, chương trình sẽ hiển thị điểm của người chơi.
- Thiết bị nhập chỉ phát hiện được nút đầu tiên được nhấn sau khi chỉ thị được đưa ra và thông báo cho chương trình về nút bấm này.

[Giải thích chương trình]

(1) Bên trong chương trình lưu giữ trạng thái hiện thời của cờ.
Trạng thái cờ đỏ: Nâng (Up) hoặc Hạ (Down)
Trạng thái cờ trắng: Nâng (Up) hoặc Hạ (Down)

(2) Chương trình sẽ đọc tệp giá trị ban đầu và thiết lập trạng thái của cờ trắng và cờ đỏ thành "Hạ (Down)"

Định dạng tệp giá trị khởi tạo

Thời gian phát hiện câu trả lời	Thời gian hiển thị cửa sổ cho trả lời sai	Số lần lặp lại

(3) Chương trình sẽ chọn ngẫu nhiên một cờ cho các chỉ lệnh nâng cờ/hạ cờ và một chỉ lệnh có thể lựa chọn dựa trên trạng thái của cờ được cho trong bảng ở dưới. Sau đó, chương trình gửi thông tin này tới các thiết bị xuất.

Bảng: Trạng thái cờ và các chỉ thị có thể lựa chọn

Trạng thái cờ	Các chỉ lệnh có thể lựa chọn	
Up	Lower	Không hạ
Down	Raise	Không nâng

(4) Có 5 kiểu thao tác có thể nhận được từ các thiết bị nhập: giương cờ đỏ, hạ cờ đỏ, giương cờ trắng, hạ cờ trắng, và không di chuyển cờ nào (khi chương trình không nhận được câu trả lời trong thời gian phát hiện câu trả lời).

(5) Chương trình sẽ phát hiện ra câu trả lời của người chơi. Nếu đúng, chương trình sẽ cộng thêm "1" vào số câu trả lời đúng và gửi cửa sổ câu trả lời đúng tới thiết bị xuất. Nếu sai, chương trình sẽ gửi cửa sổ câu trả lời sai tới các thiết bị xuất và sau khi hết thời gian hiển thị cửa sổ câu trả lời sai trong tệp giá trị khởi tạo, chương trình sẽ thiết lập cờ trong cửa sổ này sang trạng thái đúng.

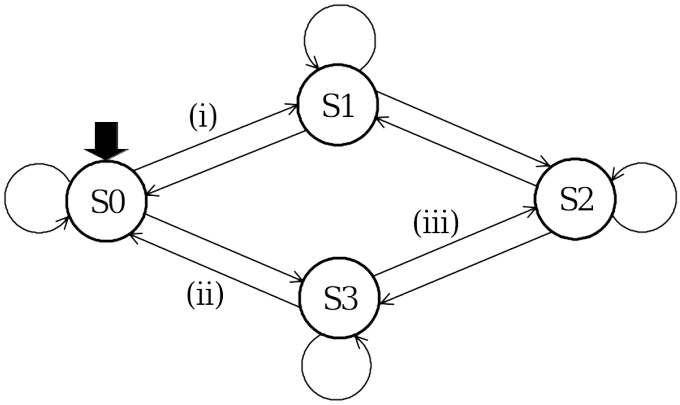
(6) Chương trình lặp lại các bước (3) tới (5) với số lần lặp lại được chỉ ra trong tệp giá trị khởi tạo.

(7) Chương trình sẽ gửi điểm của người chơi tới các thiết bị xuất.

Câu hỏi con 1

Hình 2 dưới đây là biểu đồ chuyển trạng thái của các cờ. Hãy chọn các câu trả lời đúng từ các nhóm câu trả lời dưới đây để điền vào các ô trống [ ] trong phần mô tả sau đây về Hình 2.

S0 biểu thị trạng thái khởi tạo và câu trả lời "nâng cờ đỏ (raise the red flag)" tương ứng với trạng thái (i) từ S0 tới S1. Trong trường hợp này, S1 ở trạng thái [A]. Còn câu trả lời tương ứng với bước chuyển trạng thái (ii) từ S3 tới S1 là [B] và bước chuyển trạng thái từ S3 tới S2 là [C].



Hình 2. Biểu đồ chuyển trạng thái của các cờ

Nhóm câu trả lời cho A:

a) cờ đỏ nâng và cờ trắng nâng
b) cờ đỏ nâng và cờ trắng hạ
c) cờ đỏ hạ và cờ trắng nâng
d) cờ đỏ hạ và cờ trắng hạ

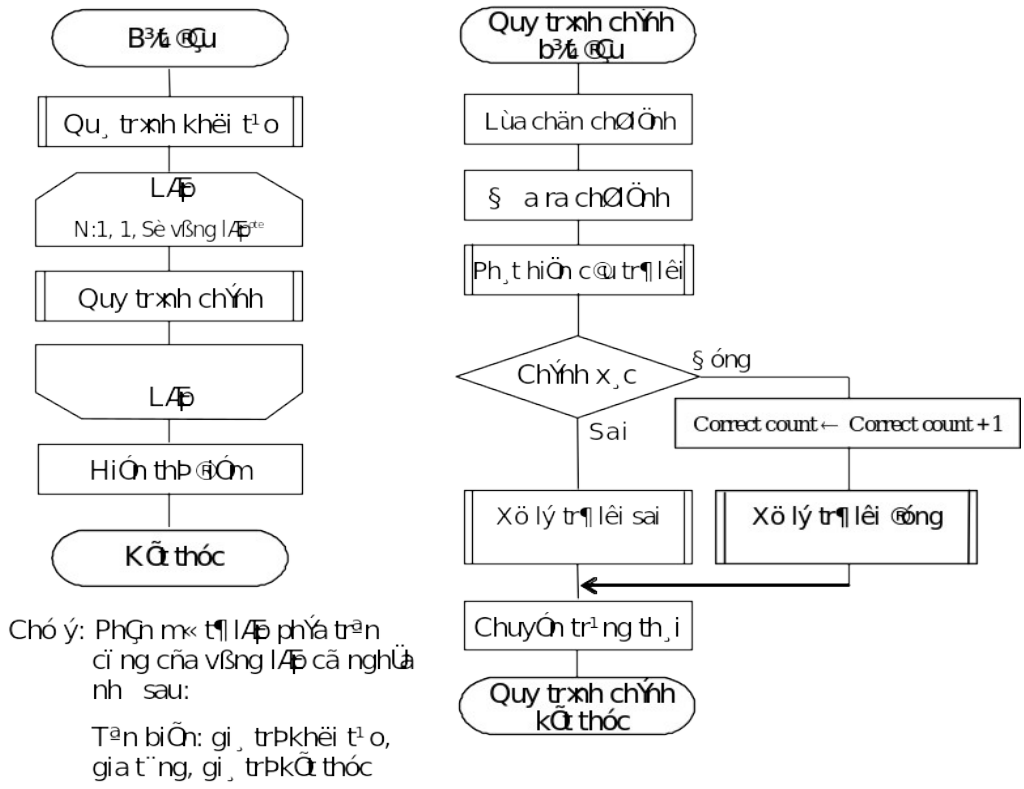
Nhóm câu trả lời cho B and C:

a) nâng cờ đỏ
b) hạ cờ đỏ
c) nâng cờ trắng
d) hạ cờ trắng

Câu hỏi con 2

Hình 3 dưới đây biểu diễn lưu đồ của chương trình này. Trạng thái ban đầu

được thiết lập trong quá trình khởi tạo. Ngoại trừ quá trình khởi tạo và thiết lập số lần lặp, **hai** quá trình nào tham chiếu tới nội dung của tệp giá trị khởi tạo?



Hình 3. Lưu đồ

Nhóm câu trả lời:

a) Xử lý trả lời sai
b) Đưa ra chỉ lệnh
c) Lựa chọn chỉ lệnh
d) Chuyển trạng thái
e) Hiển thị điểm
f) Xử lý trả lời đúng
g) Phát hiện câu trả lời

Lựa chọn một câu hỏi từ **Q6** và **Q7.** Nếu **hai câu hỏi** được lựa chọn, **chỉ câu hỏi thứ nhất** sẽ được chấm.

Q6. Đọc mô tả và đoạn chương trình C dưới đây rồi trả lời Câu hỏi con.

[Mô tả chương trình]

Chương trình này sẽ phân tách một xâu ký tự gắn thẻ và đặt các giá trị thẻ vào các phần tử riêng rẽ.

(1) Cú pháp cho các xâu ký tự gắn thẻ gồm có các ký hiệu sử dụng trong diễn giải cú pháp được định nghĩa trong Bảng 1 bên dưới và các ký hiệu <, >, / được sử dụng như các thẻ bài (token).

Bảng 1 Ý nghĩa các Ký hiệu được sử dụng trong Diễn giải Cú pháp

Ký hiệu	Ý nghĩa
::=	Được định nghĩa
ı	Hoặc
{ }*	Các phần tử bên trong "{" và "}" được lặp lại từ 0 đến nhiều lần.

tagged_character_string ::= {tagged_structure}*
tagged_structure ::= start_tag tag_value {tagged_structure}* end_tag
start_tag ::= <tag_name>
end_tag ::= </tag_name>
tag_name ::= chuỗi ký tự không bao gồm <, >, / or "\0"
tag_value ::= chuỗi ký tự không bao gồm <, >, / or "\0" ı một chuỗi ký tự trống
character_string ::= {character}*

(2) Đặc tả chương trình như sau:

(i) Hàm `parse_ml_string` sẽ phân tách một xâu tagged_character_string (thông qua con trỏ xâu `mlstr`) vào các phần tử và cất giữ chúng trong một mảng `elmtbl` theo thứ tự xuất hiện các start_tags của nó. Số các phần tử được cất giữ trong biến `elmnum`. Ở đây, mỗi phần tử được biểu diễn bởi một cấu trúc dữ liệu như sau:

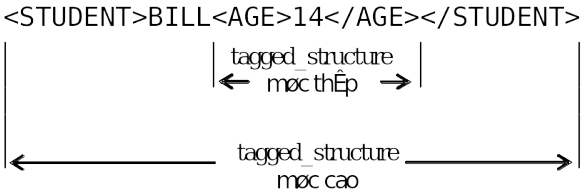
```
typedef struct {
    char *tag;     /* Tag_name */
    int  depth;    /* Độ sâu lồng nhau (1, 2,...) */
```

```
    char *value;  /* Tag_value */
} ELEMENT;
```

(ii) Số các thẻ chứa trong một xâu tagged_character_string không vượt quá 256.

(iii) Không có bất cứ lỗi cú pháp trong một xâu tagged_character_string.

(3) Một xâu tagged_structure có thể chứa một xâu tagged_structure khác. (Theo Hình 1 bên dưới.)



Hình.1 Ví dụ các tagged_structure lồng nhau

(4) Chiều sâu của các thẻ lồng nhau là các giá trị số 1, 2, 3,..., giá trị "1" được sử dụng để nói đến mức cao nhất của xâu tagged_structure.

(5) Kết quả việc thực thi chương trình này với xâu tagged_character_string được biểu diễn trong Hình 2 và Bảng 2 dưới đây.

```
<STUDENT>BILL<AGE>14</AGE><SCHOOL>Junior
<PLACE>Tokyo</PLACE></SCHOOL></STUDENT>
```

Chú ý: Xâu ký tự không bao gồm mã CR/LF.

Hình.2 Giá trị của xâu gắn thẻ mlstr

Bảng 2 Mảng `elmtbl`

Tên thẻ `tag`	Chiều sâu thẻ `depth`	Giá trị thẻ `Value`
Con trỏ trỏ tới `STUDENT`	1	Con trỏ trỏ tới `BILL`
Con trỏ trỏ tới `AGE`	2	Con trỏ trỏ tới `14`
Con trỏ trỏ tới `SCHOOL`	2	Con trỏ trỏ tới `Junior`
Con trỏ trỏ tới `PLACE`	3	Con trỏ trỏ tới

		Tokyo

[Chương trình]

```
#define MAXELMNUM 256

typedef struct {
    char *tag;
    int  depth;
    char *value;
} ELEMENT;

char *parse_ml_data(char *, int);
ELEMENT elmtbl[MAXELMNUM];
int elmnum = 0;

void parse_ml_string(char *mlstr)
{
    while (*mlstr != '\0') {
        mlstr = parse_ml_data(mlstr + 1, 1);
    }
}

char *parse_ml_data(char *mlstr, int level)
{
    /* Xử lí start_tag */
    elmtbl[elmnum].tag = [A] ;
    elmtbl[elmnum].depth = level;
    for (; *mlstr != '>'; mlstr++);
    *mlstr = '\0';
    /* Xử lí tag_value */
    elmtbl[elmnum].value = [B] ;
    for (mlstr++; *mlstr != '<'; mlstr++);
    *mlstr = '\0';
    [C] ;
    /* Xử lí tag_structure mức cao */
    while ( [D] )
        mlstr = parse_ml_data(mlstr + 1, level + 1);
    /* Xử lí End_tag */
    for (mlstr += 2; *mlstr != '>'; mlstr++);
```

```
    return mlstr + 1;
}
```

Câu hỏi con

Từ tập các câu trả lời ở dưới, lựa chọn câu trả lời chính xác chèn vào các ô trống ☐
trong chương trình ở trên.

Tập câu trả lời cho A và B:

a) `mlstr`
b) `++mlstr`
c) `mlstr + 1`
d) `*mlstr`
e) `*(++mlstr)`
f) `*mlstr++`
g) `*(mlstr + 1)`

Tập câu trả lời cho C:

a) `elmnum++`
b) `level++`
c) `mlstr++`
d) `*elmnum++`
e) `*level++`
f) `*mlstr++`

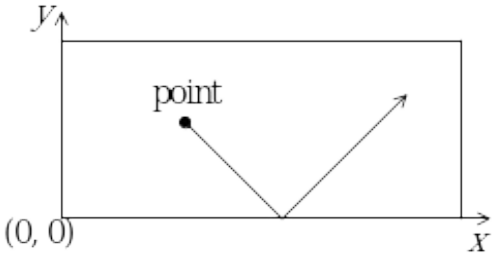
Tập câu trả lời cho D:

a) `*(mlstr + 1) == '<'`
b) `*(mlstr + 1) != '<'`
c) `*(mlstr + 1) == '/'`
d) `*(mlstr + 1) != '/'`
e) `*(mlstr + 1) == '>'`
f) `*(mlstr + 1) != '>'`

Q7. Đọc phần mô tả về một chương trình Java và chương trình đó, rồi trả lời các câu hỏi con 1 và 2.

[Mô tả chương trình]

Chương trình vẽ một điểm (point) di chuyển trong một không gian hình chữ nhật như sau:



Điểm được biểu diễn bởi lớp `Point`, lớp này lưu trữ các tọa độ (*x*□*y*) biểu diễn vị trí của điểm và vận tốc di chuyển. Vận tốc là một số dương và biểu diễn quãng đường di chuyển theo chiều trục x và trục y trên một đơn vị thời gian.
Hình chữ nhật trong hình trên được cho bởi lớp `Space`, và các phương thức lớp sau có thể được gọi.

(1) `public static int getMaxX()`
Phương thức này trả về giá trị lớn nhất (giá trị dương) của tọa độ x trong không gian hình chữ nhật.

(2) `public static int getMaxY()`
Phương thức này trả về giá trị lớn nhất (giá trị dương) của tọa độ y trong không gian hình chữ nhật

(3) `public static void draw(Point)`
Phương thức này vẽ điểm được xác định bởi đối số tại các tọa độ của điểm.

(4) `public static void erase(Point)`
Phương thức này dùng để xóa điểm được xác định bởi đối số.

Lớp trừu trượng `Motion` sử dụng một thể hiện điểm của lớp `Point` làm giá trị ban đầu trong phương thức khởi tạo, vẽ,di chuyển và xóa điểm lặp đi lặp lại theo thứ tự đó để hiển thị sự chuyển động của điểm đó. Các tọa độ của điểm sau khi di chuyển được đưa ra bởi phương thức `update`.
`SimpleMotion` là lớp con của lớp `Motion`, nó cài đặt phương thức `update` và `main`. Phương thức `update` biểu diễn chuyển động của một điểm theo đường thẳng với vận tốc không đổi trong không gian hình chữ nhật và nảy ra khỏi các cạnh của hình chữ nhật. Phương thức `main` dùng để chạy thử chương trình.

Chú ý rằng giả sử các giá trị ban đầu của các tọa độ của `Point` được sinh ra bởi phương thức `main` nằm trong không gian hình chữ nhật, và không cần tính đến các xung đột giữa các điểm

[Chương trình 1]

```
public class Point {
    private int x, y, speed;
    public Point(int x, int y, int speed) {
        this.x = x; this.y = y;
        this.speed = speed;
    }
    public int getX() { return x; }
    public int getY() { return y; }
    public int getSpeed() { return speed; }
}
```

[Chương trình 2]

```
public abstract class Motion implements Runnable {
    private Point point;

    public Motion(Point point) {
        this.point = point;
    }

    public void run() {
        while (true) {
            Space.draw(point);
            try {
                Thread.sleep(40);
            } catch (InterruptedException e) {}
            Point current = point;
            [          A          ];
            Space.erase(current);
        }
```

```
    }

    public abstract Point update(Point point);
}
```

[Chương trình 3]

```
public class SimpleMotion extends Motion {
    private int directionX = 1, directionY = 1;

    public SimpleMotion(Point point) {
        super(point);
    }

    public Point update(Point point) {
        int speed = point.getSpeed();
        int x = point.getX() + directionX * speed;
        int y = point.getY() + directionY * speed;
        if (x <= 0) {
            x = -x;
            directionX *= -1;
        }
        x %= 2 * Space.getMaxX();
        if (x >= Space.getMaxX()) {
            x = 2 * Space.getMaxX() - x;
            directionX *= -1;
        }
        if (y <= 0) {
            y = -y;
            directionY *= -1;
        }
        y %= 2 * Space.getMaxY();
        if (y >= Space.getMaxY()) {
            y = 2 * Space.getMaxY() - y;
            directionY *= -1;
        }
        return new Point(x, y, speed);
    }
```

```
    public static void main(String[] args) {
        Point[] points = {
            new Point(10,  20, 3),
            new Point(50,  10, 5),
            new Point(150, 60, 2)
        };
        for (int i = 0; i < points.length; i++) {
            new Thread(          B          ).start();
        }
    }
}
```

Câu hỏi con 1

Từ các nhóm câu trả lời dưới đây, chọn các câu trả lời đúng để điền vào các ô trống ☐ trong các chương trình trên.

Nhóm câu trả lời cho A:

a) `current = update(current)`
b) `current = update(point)`
c) `point = update(point)`
d) `update(current)`
e) `update(point)`

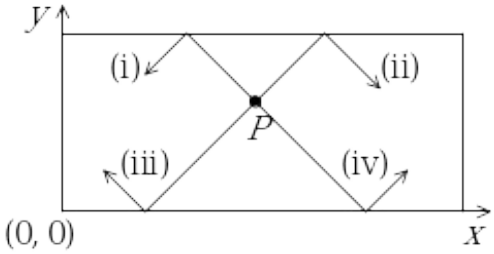
Nhóm câu trả lời cho B:

a) `new Motion(points[i])`
b) `new Point(points[i])`
c) `new Runnable(points[i])`
d) `new SimpleMotion(points[i])`
e) `points[i]`

Câu hỏi con 2

Từ nhóm câu trả lời dưới đây, chọn câu trả lời đúng cho chuyển động của điểm *P* trong hình dưới đây khi phương thức `run` được thực thi. Giả sử rằng điểm P là một thể hiện của lớp `Point` được xác định cho phương thức khởi tạo của lớp `SimpleMotion`, và giá trị của `speed` là 1.

Biết rằng tất cả các ☐ trong chương trình trên đều có các câu trả lời đúng.



Nhóm câu trả lời:

a) Điểm P chuyển động theo hình mũi tên (i).
b) Điểm P chuyển động theo hình mũi tên (ii).
c) Điểm P chuyển động theo hình mũi tên (iii).
d) Điểm P chuyển động theo hình mũi tên (iv)

Lựa chọn một câu hỏi từ **Q8** và **Q9.** Nếu **hai câu hỏi** được lựa chọn, **chỉ câu hỏi thứ nhất** sẽ được chấm.

Q8. Đọc mô tả chương trình và chương trình C dưới đây rồi trả lời các câu hỏi con từ 1 đến 4.

[Mô tả chương trình]

Chương trình tìm đường qua một mê cung.

(1) Dữ liệu cho mê cung được hiển thị bằng một hình vuông 8 cột và 8 dòng, như được chỉ ra trong Hình 1 dưới đây, được lưu trong một mản `M` hai chiều. Các hình vuông được tô màu xám biểu thị tường, màu trắng biểu thị đường đi.

(2) Mảng `M` lưu trữ một mã biểu thị điểm bắt đầu (0xf0), được gán cho hình vuông lối vào, một mã biểu thị điểm kết thúc (0xf1), một mã biểu thị cho tường (0x00), được gán cho các hình vuông tường, và một mã biểu thị cho đường đi (0xff), được gán cho các hình vuông đường đi.

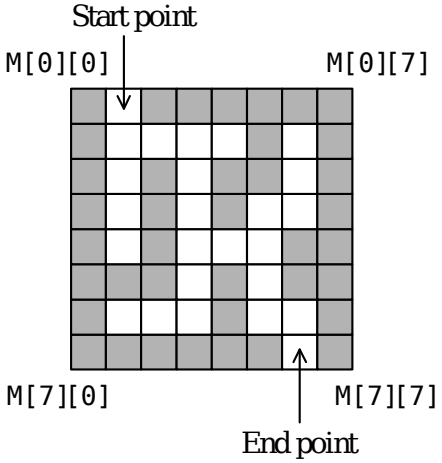


Fig. 1 Example of Maze

(3) Ngoại trừ điểm bắt đầu và điểm kết thúc, tất cả các hình vuông trên viền ngoài của mê cung (các hình vuông có giá trị cột hoặc hàng bên trong `M` hoặc “0” hoặc “7”) là tường.

(4) Dưới đây là các biến toàn cục được sử dụng trong chương trình này:

Tên biến	Mục đích
`M`	Lưu trữ dữ liệu mê cung (mảng hai chiều)
`x`	Hàng của hình vuông bên trong mê cung

`y`	Cột của hình vuông bên trong mê cung
`dir`	Hướng di chuyển trong khi đi qua mê cung

Giả sử rằng các giá trị khởi tạo được nạp vào các biến toàn cục này.

Hàm `maze` tìm kiếm qua mê cung để tìm đường từ lối vào đến lối ra.

[Program]

(Line No.)

```
 1 #define   UP          0
 2 #define   RIGHT       1
 3 #define   DOWN        2
 4 #define   LEFT        3
 5 #define   ROAD        0x00        /* Mã đường đi */
 6 #define   WALL        0xff        /* Mã tường */
 7 #define   SMAX        8
 8 #define   ENTRANCE    0xf0        /* Mã điểm bắt đầu */
 9 #define   EXIT        0xf1        /* Mã điểm kết thúc */
10
11 int rcheck(void);
12 int fcheck(void);
13 void go(void);
14 void maze(void);
15
16 int M[SMAX][SMAX], x, y, dir;
17
18 void maze()
19 {
20      while ( M[y][x] != EXIT ) {
21          if ( ( rcheck() == ROAD ) ||
22              ( rcheck() == EXIT ) ) {
23              dir = ( dir+1 ) % 4;
24              go();
25          }
26          else if ( ( fcheck() == ROAD ) ||
27                    ( fcheck() == EXIT ) ) go();
28          else dir = ( dir+3 ) % 4;
29      }
30      return;
31 }
32
33 int rcheck()
34 {
35      if      ( dir == UP )      return M[y][x+1];
```

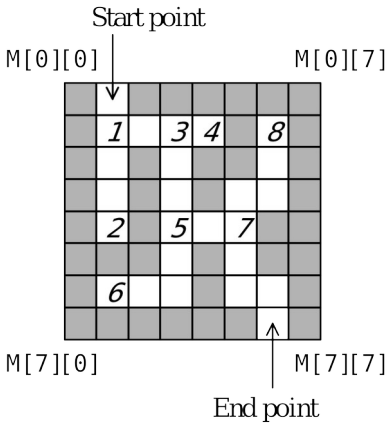
```
36      else if ( dir == RIGHT )   return M[y+1][x];
37      else if ( dir == DOWN )    return M[y][x-1];
38      else                       return M[y-1][x];
39 }
40
41 int fcheck()
42 {
43      if      ( dir == UP )      return M[y-1][x];
44      else if ( dir == RIGHT )   return M[y][x+1];
45      else if ( dir == DOWN )    return M[y+1][x];
46      else                       return M[y][x-1];
47 }
48
```

```
49 void go()
50 {
51      if      ( dir == UP )      y--;
52      else if ( dir == RIGHT )   x++;
53      else if ( dir == DOWN )    y++;
54      else                       x--;
55 }
```

Câu hỏi con 1

Cho mê cung được chỉ ra trong Hình 2 ở dưới, đâu là kết quả đường đầu ra của bởi chương trình này? Lựa chọn câu trả lời đúng trong nhóm câu trả lời bên dưới. *1* đến *8* được chỉ ra trong hình được sử dụng để biểu diễn vị trí các hình vuông trong mê cung. Trong chương trình này, giá trị các biến toàn cục `x`, `y`, và `dir` được chỉ ra như dưới đây tại thời điểm hàm `maze` được gọi:

```
x = 1
y = 0
dir = DOWN
```



Hình. 2 Các vị trí của hình vuông trong mê cung

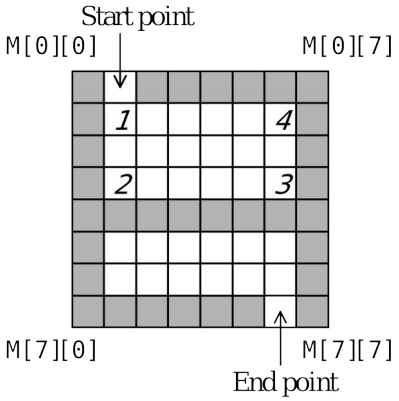
Nhóm câu trả lời:

a) Điểm bắt đầu→*1*→*2*→*1*→*3*→*5*→*6*→*5*→*7*→Điểm kết thúc
b) Điểm bắt đầu→*1*→*2*→*1*→*3*→*4*→*3*→*5*→*6*→*5*→*7*→*8*→*7*→Điểm kết thúc
c) Điểm bắt đầu→*1*→*3*→*5*→*7*→Điểm kết thúc
d) Điểm bắt đầu→*1*→*3*→*4*→*3*→*5*→*6*→*5*→*7*→*8*→*7*→Điểm kết thúc

Câu hỏi con 2

Cho mê cung trong Hình 3 dưới đây, đâu là kết quả đường đầu ra bởi chương trình này? Lựa chọn câu trả lời đúng trong nhóm câu trả lời phía dưới. Trong chương trình này, giá trị của các biến toàn cục `x`, `y`, và `dir` được chỉ ra như dưới đây tại thời điểm hàm `maze` được gọi.

```
x = 1
y = 0
dir = DOWN
```



Hình. 3 Mê cung không giải quyết được

Nhóm câu trả lời:

a) Điểm bắt đầu→*1*→*4*→*3*→*2*→*1*→*4*→*3*→*2*→... được lặp lại vô hạn.
b) Điểm bắt đầu→*1*→*2*→*3*→*4*→*1*→*2*→*3*→*4*→... được lặp lại vô hạn.
c) Điểm bắt đầu→*1*→*2*→*2*→*2*→... được lặp lại vô hạn.
d) Điểm bắt đầu→*1*→*2*→*3*→*4*→*1*→ Điểm bắt đầu (Chương trình kết thúc)
e) Điểm bắt đầu→*1*→*2*→*3*→*4*→*1*→*1*→*1*→... được lặp lại vô hạn.

Câu hỏi con 3

Câu lệnh sau đây được thêm vào chương trình để đưa ra các vị trí của các hình vuông, bao gồm điểm bắt đầu và điểm kết thúc, được đi qua và các vị trí của các di chuyển. Đâu là chỗ đúng nhất để thêm vào câu lệnh này trong chương trình? Hãy chọn câu trả lời đúng từ nhóm câu trả lời phía dưới.
Giả sử rằng `#include <stdio.h>` được đặt ở đầu của chương trình.

```
printf("dir=%d y=%d x=%d\n", dir, y, x);
```

Nhóm câu trả lời:

a) Ngay sau Dòng 20 và ngay sau Dòng 29
b) Ngay sau Dòng 23 và ngay sau Dòng 29
c) Ngay sau Dòng 28 và ngay sau Dòng 29

Câu hỏi con 4

Giả sử rằng hàm `lcheck` được sử dụng thay cho `rcheck` để tìm ra các giải pháp cho mê cung, và hàm `lcheck` thay cho `rcheck` trong Dòng 11, 21, và 22.
Nếu sự thay đổi trên được thực hiện thì cần phải thay đổi những gì cho chương trình? Hãy chọn câu trả lời đúng từ nhóm câu trả lời dưới đây.

```
int lcheck()
{
    if      ( dir == UP )       return M[y][x-1];
    else if ( dir == RIGHT )return M[y-1][x];
    else if ( dir == DOWN ) return M[y][x+1];
    else                    return M[y+1][x];
}
```

Nhóm câu trả lời:

a) `== ROAD` trên Dòng 21 phải được thay bởi `== WALL`.
b) `+1` trên Dòng 23 phải được thay đổi thành `+3` và `+3` ở Dòng 28 phải được thay đổi thành `+1`.
c) `UP` trên Dòng 35 và 43 phải được thay đổi thành `RIGHT`, `RIGHT` trên Dòng 36 và 44 phải được thay đổi thành `DOWN`, và `DOWN` trên Dòng 37 và 45 phải được thay đổi thành `LEFT`.
d) Tất cả các thể hiện của — từ Dòng 51 đến Dòng 54 phải được

thay đổi thành ++ và tất cả các thể hiện của ++ được thấy trên cùng các dòng ấy phải được thay đổi thành --.

Q9. Đọc mô tả chương trình và chương trình Java dưới đây rồi trả lời câu hỏi con 1 và 2.

[Mô tả chương trình]

Đây là chương trình cho một máy tính điện tử thực hiện các phép toán cộng, trừ, nhân và chia trên các số nguyên. Thành phần I/O được cung cấp bởi một chương trình chạy thử, và có thể được sử dụng để chạy thử chương trình thành phần máy tính chính.

(1) Lớp `CalculatorEvent` là một sự kiện được sinh ra khi một nút trên máy tính được ấn. Giá trị của trường `type` biểu diễn các sự kiện. Các sự kiện có thể là `DIGIT`, `OPERATOR`, hoặc `CLEAR`, và chúng biểu diễn tương ứng cho các nút số (từ 0 đến 9) trên máy tính, nút phép toán (ví dụ như +) hoặc nút bằng (=), hoặc nút xóa (C). Khi `type` là `DIGIT`, giá trị số tương ứng với nút số được lưu trong trường `value`. Khi `type` là `OPERATOR`, kí tự biểu diễn cho loại phép toán hoặc '=' được lưu trong trường `value`. Khi `type` là is `CLEAR` thì `value` không được sử dụng.

(2) Giao diện `CalculatorOutput` khai báo phương thức `display` hiển thị các giá trị số và lỗi trên máy tính.

(3) Lớp `Calculator` chính là máy tính chính.
Phương thức `eventDispatched` nhận các sự kiện và thực hiện các phép toán,... phù hợp với loại sự kiện.
Chú ý rằng các kết quả tính toán – cộng, trừ, nhân và chia – của hai giá trị số phù hợp với kết quả của các phép toán đó trên kiểu dữ liệu `int` của Java.

(4) Lớp `CalculatorTest` là một chương trình để chạy thử `Calculator`.
`CalculatorOutput` được cài đặt là một lớp không tên. Trong cài đặt này, phương thức `display` đưa ra giá trị số hoặc chuỗi kí tự bởi `System.out`.
Phương thức `main` sinh ra `CalculatorEvent` từ chuỗi kí tự được lấy về từ tham số `args[0]`, và gọi phương thức `eventDispatched` của `Calculator`.
Sự tương ứng giữa các kí tự và các nút trên máy tính được chỉ ra trong bảng dưới đây.

Kí tự	Nút trên máy tính
'0' đến '9'	Nút số (0 đến 9)
'+'	Nút cộng (+)
'-'	Nút trừ (-)
'*'	Nút nhân (×)

'/'	Nút chia (÷)
'='	Nút bằng (=)
'C'	Nút xóa (C)

Ví dụ như, chuỗi kí tự “2+7=” biểu diễn các nút trên máy tính theo thứ tự 2, +, 7, và = đang được bấm. Khi chuỗi kí tự được truyền cho phương thức `main` cho tham số `args[0]`, chương trình hiển thị kết quả như sau:

```
2
2
7
9
```

[Chương trình 1]

```
public class CalculatorEvent {
    public static final int DIGIT = 1;
    public static final int OPERATOR = 2;
    public static final int CLEAR = 3;

    private int type, value;

    public CalculatorEvent(int type) {
        [          A          ];
    }
    public CalculatorEvent(int type, int value) {
        if (type < DIGIT || type > CLEAR)
            throw new IllegalArgumentException();
        this.type = type; this.value = value;
    }
    public int getType() { return type; }
    public int getValue() { return value; }
}
```

[Chương trình 2]

```
public interface CalculatorOutput {
    public void display(int value);
```

```
    public void display(String value);
}
```

[Chương trình 3]

```
public class Calculator {
    private int accumulator = 0, register = 0;
    private int operator = 0;
    private CalculatorOutput output;

    public Calculator(CalculatorOutput output) {
        this.output = output;
    }

    public void eventDispatched(CalculatorEvent event) {
        switch (event.getType()) {
        case CalculatorEvent.DIGIT:
            if (operator == '=') {
                register = 0; operator = 0;
            }
            register = register * 10 + event.getValue();
            output.display(register);
            break;
        case CalculatorEvent.OPERATOR:
            try {
                register = calculate();
                output.display(register);
                accumulator = register;
                operator = event.getValue();
            } catch (ArithmeticException e) {
                output.display("Error");
                accumulator = 0; operator = 0;
            }
            if (operator != '=')
                register = 0;
            break;
        case CalculatorEvent.CLEAR:
            register = 0;
            accumulator = 0;
            operator = 0;
            output.display(register);
```

```
            break;
        }
    }
    private int calculate() {
        switch (operator) {
        case '+':
            return accumulator + register;
        case '-':
            return accumulator - register;
        case '*':
            return accumulator * register;
        case '/':
            return accumulator / register;
        }
        return register;
    }
}
```

[Chương trình 4]

```
public class CalculatorTest {
    public static void main(String[] args) {
        Calculator calc = new Calculator(
            [          B          ] {
                public void display(int value) {
                    System.out.println(value);
                }
                public void display(String value) {
                    System.out.println(value);
                }
            });
        String keys = args[0];
        for (int i = 0; i < keys.length(); i++) {
            char c = keys.charAt(i);
            CalculatorEvent event = null;
            if (c ≥ '0' && c ≤ '9') {
                event = new CalculatorEvent(
                                [          C          ]);
```

```
            } else if (c == '=' || c == '+' || c == '-'
                        || c == '*' || c == '/') {
                event = new CalculatorEvent(
                        CalculatorEvent.OPERATOR, c);
            } else if (c == 'C') {
                event = new CalculatorEvent(
                        CalculatorEvent.CLEAR);
            }
            if (event != null)
                calc.eventDispatched(event);
        }
    }
}
```

Câu hỏi con 1

Từ các nhóm câu trả lời dưới đây, hãy chọn các câu trả lời đúng để điền vào các ô trống [ ] trong các chương trình trên.

Nhóm câu trả lời cho A:

a) `CalculatorEvent(type, 0)`
b) `new CalculatorEvent(type, 0)`
c) `return new CalculatorEvent(type, 0)`
d) `super(type, 0)`
e) `this(type, 0)`

Nhóm câu trả lời cho B:

a) `implements CalculatorOutput()`
b) `interface CalculatorOutput()`
c) `new CalculatorOutput()`
d) `new Temp() implements CalculatorOutput`
e) `public class Temp implements CalculatorOutput`

Nhóm câu trả lời cho C:

a) `c - '0', CalculatorEvent.DIGIT`
b) `c, CalculatorEvent.DIGIT`
c) `CalculatorEvent.DIGIT`
d) `CalculatorEvent.DIGIT, c`
e) `CalculatorEvent.DIGIT, c - '0'`

Câu hỏi con 2

Bảng sau đây chỉ ra các kết quả đầu ra cuối cùng khi phương thức `main` được thực thi với các xâu kí tự ở dưới được truyền vào như các tham số.

Từ các nhóm câu trả lời dưới đây, hãy chọn câu trả lời đúng cần chèn vào ô trống [ ] trong bảng.

Giả thiết rằng tất cả các ô trống [ ] trong chương trình đều có câu trả lời đúng.

Xâu kí tự	Kết quả
3+4*5=	35
3*4***=	[D]
3*4=+5	[E]
3+4/0=	[F]

Nhóm câu trả lời:

a) `0` b) `3` c) `4` d) `5`
e) `7` f) `12` g) `17` h) `53`
i) `/ by zero` j) `Error`

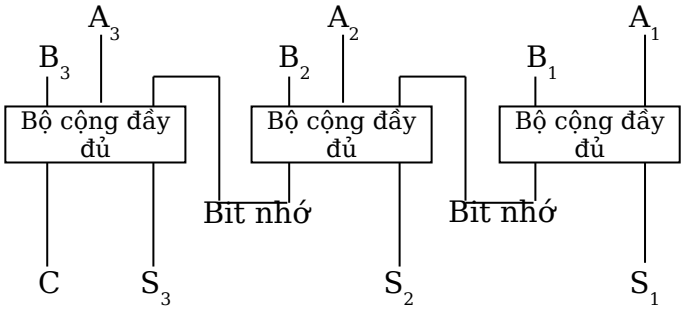
Thi thử Đáp án và giải thích cho các câu hỏi buổi chiều

Q1:

Điểm chính	➢ **Bộ bán tổng có 2 đầu vào trong khi bộ cộng đầy đủ có 3 đầu vào.** ➢ **Phép cộng logic loại trừ là phép toán mà nhận giá trị 0 khi 2 giá trị đầu vào có giá trị giống nhau.**

Bộ cộng là một mạch dùng để cộng các bit nhị phân với nhau, có thể là bộ bán tổng hoặc bộ cộng đầy đủ. Bộ bán tổng là mạch mà không quan tâm đến nhớ ở chữ số thấp hơn (bit thấp hơn) và được sử dụng để cộng chữ số thấp nhất (bit thấp nhất). Bộ cộng đầy đủ là mạch mà tính cả nhớ từ các chữ số thấp hơn và được sử dụng để cộng các chữ số không phải chữ số thấp nhất

Hình dưới đây, đưa một ví dụ về cộng số 3 bit. A_i (i = 1, 2, 3) và B_i (i = 1, 2, 3) là các con số nhị phân được cộng. S_i (i = 1, 2, 3) là kết quả của phép cộng mỗi bit và C là bit nhớ được dùng để cộng các bit cao hơn



$$\begin{array}{r} A_3A_2A_1 \\ +)\ B_3B_2B_1 \\ \hline CS_3S_2S_1 \end{array}$$

Câu hỏi con 1: [Câu trả lời đúng] c

C_k (= 1), A_k (= 0), và B_k (= 1) được cộng cùng nhau, và kết quả của phép cộng này là Z_k với một số nhớ C_{k+1}. Phép cộng được thực hiện như sau:

$$\begin{array}{r} 1 = C_k \\ +)\ 0 = A_k \\ \hline 1 \end{array} \longrightarrow \begin{array}{r} 1 = C_k + A_k \\ +)\ 1 = B_k \\ \hline 10 \end{array}$$

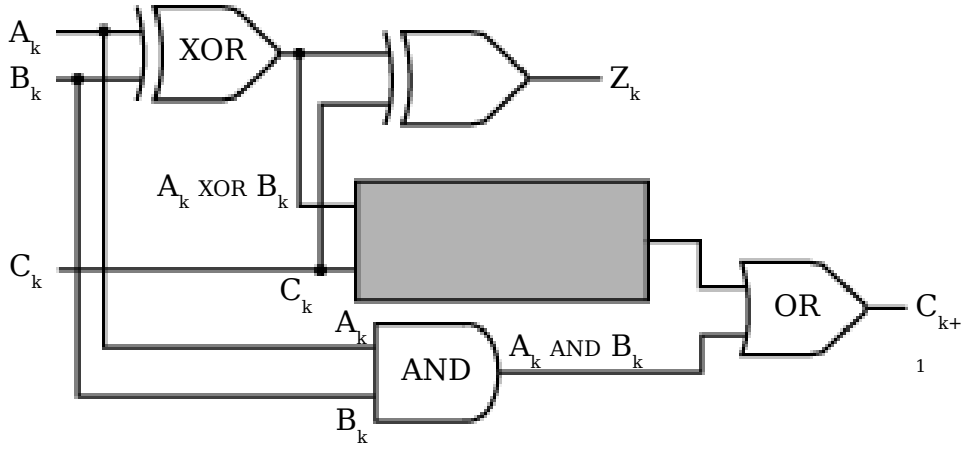
C_{k+1} Z_k

Vì C_{k+1} = 1 và Z_k = 0, chúng ta có như sau:

Đầu vào			Đầu ra	
C_k	A_k	B_k	C_{k+1}	Z_k
1	0	1	1	0

Câu hỏi 2: [Câu trả lời đúng] a

Phép toán để lấy được C_{k+1} như sau:



Như trong hình này, nếu chúng ta để P là kết quả của phép logic trong hộp màu xám, thì kết quả của phép cộng logic (OR) của P và "A_k AND B_k" là C_{k+1}. Thêm vào đó, P là kết quả của các phép toán nhị phân "A_k XOR B_k" và C_k. Từ nhóm các câu trả lời, chỉ ra rằng phép toán này là phép nhân logic (AND), phép tổng logic (OR) hoặc phép cộng logic loại trừ (XOR). Vì vậy chúng ta có thể tổ chức toàn bộ thông tin trong bảng sau.

C_k	A_k	B_k	P	A_k AND B_k	(P OR A_k AND B_k) = C_{k+1}
0	0	0	?	0	0
0	0	1	?	0	0
0	1	0	?	0	0
0	1	1	?	1	1
1	0	0	?	0	0
1	0	1	?	0	1
1	1	0	?	0	1
1	1	1	?	1	1

Ở đây, tổng logic của P và "A_k AND B_k" là C_{k+1}. Do đó chúng ta của thể "ước lượng" được giá trị của P như sau. Vì phép toán ở đây là phép cộng logic, nếu C_{k+1} là "0", thì cả hai biến (đầu vào) đều phải bằng 0. Nhưng nếu C_{k+1} là 1, thì một trong hai biến đầu vào sẽ bằng 1.

A_k AND B_k	$C_k + 1$	Giá trị ước lượng của P
0	0	0
0	0	0
0	0	0
1	1	1, 0
0	0	0
0	1	1
0	1	1
1	1	1, 0

Từ bảng ước lượng giá trị của P, chúng ta có thể biết được phép toán giữa "A_k

XOR B_k" và C_k. Hơn nữa từ nhóm câu trả lời chỉ có các phép nhân logic, cộng logic và cộng loại trừ nên chúng ta chỉ giới hạn các câu trả lời trong đó.

A_k XOR B_k	C_k	P	Các phép toán logic có thể
0	0	0	Nhân logic, Cộng logic, Cộng loại trừ
1	0	0	Nhân logic
1	0	0	Nhân logic
0	0	1, 0	Nhân logic nếu là 0. Không có phép toán nào nếu là 1.
0	1	0	Nhân logic
1	1	1	Nhân logic, Cộng logic
1	1	1	Nhân logic, Cộng logic
0	1	1, 0	Nhân logic nếu là 0. Cộng logic hoặc cộng logic loại trừ nếu là 1

Từ đó chúng ta thấy rằng chỉ có một phép toán thỏa mãn tất cả các trường hợp đó là phép nhân logic

Câu hỏi 3: [Câu trả lời đúng]

Phần màu xám trong bảng thật có giá trị như dưới đây. Trong câu hỏi, chỉ số là k, nhưng vì chúng ta chỉ quan tâm đến bit cao nhất (có ý nghĩa lớn nhất), nên chúng ta sử dụng chỉ số n. Bit cao nhất là C_n gợi ý rằng thanh ghi có n bit, do đó C_{n+1} trở thành bit tràn, không nằm trong thanh ghi.

	Input			Output	
	C_k	A_k	B_k	C_{k+1}	Z_k
Case 1	0	1	1	1	0
Case 2	1	0	0	0	1

Dưới đây, chúng ta cộng C_n, A_n, và B_n trong 2 trường hợp.

(1) trường hợp 1

$0 = C_n$
$1 = A_n$
$+) \ 1 = B_n$
10
$C_{n+1} \ Z_n$

Trong trường hợp này, A_n và B_n là các bit cao nhất, cả hai đều có giá trị 1, tức là chúng ta đang cộng 2 số âm. Tuy nhiên Z_n bit cao nhất của kết quả là 0 chỉ ra rằng đó là một số dương. Do đó phép toán đã không được thực hiện chính xác. Điều này là vì đã có hiện tượng tràn số xảy ra, đẩy giá trị 1 chỉ ra rằng đó là số âm cho C_{n+1} (không xuất hiện trong kết quả)

(2) Trường hợp 2

$1 = C_n$
$0 = A_n$
$+) \ 0 = B_n$
01
$C_{n+1} \ Z_n$

Ở đây, A_n và B_n là các bit cao nhất, cả hai đều mang giá trị 0, vì thế nó là phép cộng hai số dượng. Tuy nhiên Z_n bít cao nhất của kết quả lại mang giá trị 1, chỉ ra rằng đó là một số âm. Do đó phép toán cũng không được thực hiện chính xác. Điều này là vị hiện tượng tràn số bị xảy ra, đẩy giá trị 0 chỉ ra rằng đây là số dương cho C_{n+1} (không xuất hiện trong kết quả)

Chúng ra cùng tổng hợp kết quả:

A_n	B_n	C_n	Z_n	C_{n+1}
0	0	1	1	0
1	1	0	0	1

Bởi vì phép tràn xảy ra với bit nhớ, nên chúng ta chỉ quan tâm đến C_n và C_{n+1}. Sau đó chúng ta để ý rằng tràn xảy ra khi hai giá trị này không bằng nhau. Trong câu hỏi con, đầu tiên phép cộng logic loại trừ được thực sử dụng, và phép tổng logic loại trừ của 2 giá trị C_n và C_{n+1} là 1 khi chúng không bằng nhau. Do đó việc tràn có thể được phát hiện bằng cách cộng logic loại trừ C_n và C_{n+1}.

Q2:

Điểm chính	➢ **EXISTS trả về "true" khi kết quả của câu truy vấn con có ít nhất 1 dòng.** ➢ **DISTINCT loại bỏ các trùng lặp.**

EXISTS là một mệnh đề SQL được thực thi từ câu truy vấn con (ngoài mệnh đề SELECT). Với mỗi dòng kết quả của câu truy vấn chính, nếu kết quả của câu truy vấn con (mệnh đề SELECT trong ngoặc đơn) có ít nhất 1 dòng, giá trị "true" được trả về; nếu không có dòng nào thì giá trị "false" được trả về. Nếu giá trị là "true", nội dung của cột được chỉ trong mệnh đề SELECT trong câu truy vấn chính được lựa chọn.

Câu hỏi con 1: [Đáp án đúng] A – g, B – c

A:
Trong mệnh đề SELECT ở câu truy vấn con, điều kiện lựa chọn dữ liệu là "skill_code = 'FE'. Trong bảng kỹ năng nhân viên có 2 dòng mà mã kỹ năng là "FE", vì thế EXISTS trả về "true". Do vậy câu lệnh SQL có thể được làm rõ như sau:

SELECT MaNhanVien **FROM** Bang_Ky_Nang_Nhan_Vien **WHERE** (dieu_kien = true)

Vì vậy chúng ta có thể kết luận rằng vì tất cả các dòng trong bảng kỹ năng nhân viên đều thỏa mãn điều kiện nên tất cả các mã nhân viên từ bảng kỹ năng đều được lựa chọn. Bảng này có 6 dòng.

B:
Mệnh đề SELECT trong câu truy vấn chính lấy một dòng từ bảng nhân viên, giá trị của nó được chuyển đến câu truy vấn con. Trong câu truy vấn con, nội dung của dòng nhận được được đánh giá và lấy kết quả trả về cho câu truy vấn chính. Sau đó, câu truy vấn chính đánh giá kết quả của câu truy vấn con liên quan đến dòng đã chọn trước đó rồi quyết định xem có lấy làm kết quả cuối cùng hay không.
Trong câu truy vấn con, bảng nhân viên (A) và bảng kỹ năng nhân viên (B) được kết nối và các hàng có mã kỹ năng là "FE" được chọn. Ví dụ, dòng đầu tiên của bảng nhân viên (0001, Brown, A1) được kết nối với bảng kỹ năng nhân viên bởi nhân viên mã "0001" như hình dưới đây:

Bang_Nhan_Vien

MaNhanVien	TenNhanVien	Phong
0001	Brown	A1
0002	Charles	A2
0003	Taylor	B1
0004	Williams	D3
0005	Parker	A1
0006	James	B1

Bang_Ky_Nang_Nhan_Vien

MaNhanVien	MaKyNang	NgayDangKy
0001	FE	19991201
0001	DB	20010701
0002	SAD	19980701
0002	FE	19990701
0002	SW	20000701
0005	SAD	19991201

Bây giờ có 2 dòng trong bảng kỹ năng nhân viên có cùng một mã nhân viên "001" nhưng chỉ một dòng đầu tiên trong đó có mã kỹ năng là "FE". Ở đây, kết quả của câu truy vấn con là "true" (có 1 dòng) vì thế "TenNhanVien" xác định trong câu truy vấn chính được chọn từ bảng nhân viên. Tương tự, dòng thứ 2 của bảng nhân viên (0002, Charles, A2) có mã nhân viên xuất hiện trong các dòng 3,4,5 của bảng kỹ năng nhưng chỉ có dòng 4 có mã kỹ năng "FE". Do đó tên nhân viên này được chọn. Ngược lại, không có dòng nào trong bảng kỹ năng có mã nhân viên giống các dòng 3,4 hoặc 6 của bảng nhân viên. Do đó các tên nhân viên này không được chọn. Với dòng 5 (0005, Parker, A1) của bảng nhân viên, có một dòng trong bảng kỹ năng có cùng mã nhân viên, tuy nhiên mã kỹ năng khác "FE" nên tên nhân viên này không được chọn.
Tóm lại, chúng ta thấy có 2 dòng (Brown và Charles) được chọn như hình vẽ dưới đây.
Trong hình vẽ này các dòng nét liền chỉ ra các mối liên hệ mà các đối tượng được chọn; các dòng nét đứt chỉ ra các mã nhân viên có tồn tại trong bảng kỹ năng nhưng mã kỹ năng khác "FE".

Bang_Nhan_Vien

MaNhanVien	TenNhanVien	Phong
0001	Brown	A1
0002	Charles	A2
0003	Taylor	B1
0004	Williams	D3
0005	Parker	A1
0006	James	B1

Bang_Ky_Nang_Nhan_Vien

MaNhanVien	MaKyNang	NgayDangKy
0001	FE	19991201
0001	DB	20010701
0002	SAD	19980701
0002	FE	19990701
0002	SW	20000701
0005	SAD	19991201

Ý tưởng của EXISTS đã được thể hiện ở trên. Tuy nhiên kết quả cuối cùng ở đây là những tên nhân viên nào thuộc các hàng thỏa mãn điều kiện câu truy vấn con (MaKyNang=”FE”) được chọn.

Câu hỏi con 2: [Đáp án đúng] d

Trong mệnh đề SQL này bảng kỹ năng nhân viên được định nghĩa bởi 2 tên B1 và B2. Kết quả sẽ như sau: điều kiện của câu truy vấn con "B1. MaNhanVien=B2.MaNhanVien" tìm các hàng với cùng một mã nhân viên trong bảng kỹ năng. Hơn nữa, trong các hàng này nếu thỏa mãn điều kiện khác mã kỹ năng (B1.MaKyNang <> B2.MaKyNang) sẽ được chọn.
Tóm lại chúng ta có thể vẽ một hình vẽ như dưới đây. Các dòng nét liền chỉ ra chúng được chọn vì có mã kỹ năng khác nhau; các dòng nét đứt chỉ ra chúng không được chọn vì có mã kỹ năng giống nhau.

Bang_Ky_Nang_Nhan_Vien (B1)

MaNhanVien	MaKyNang	NgayDangKy
0001	FE	19991201
0001	DB	20010701

Bang_Ky_Nang_Nhan_Vien (B2)

MaNhanVien	MaKyNang	NgayDangKy
0001	FE	19991201
0001	DB	20010701

		1
0002	SAD	19980701
0002	FE	19990701
0002	SW	20000701
0005	SAD	19991201

0002	SAD	19980701
0002	FE	19990701
0002	SW	20000701
0005	SAD	19991201

Nếu không bận tâm đến DISTINCT, *vì các mã nhân viên của các hàng có đường nét liền tương ứng được đưa ra khỏi B1,* các giá trị được chọn là (0001, 0001, 0002, 0002, 0002, 0002, 0002, 0002). Tuy nhiên mệnh đề SELECT trong câu truy vấn chính đưa ra DISTINCT nên các giá trị trùng lặp được loại bỏ và tuần tự (0001, 0002) được chọn.
Vì vậy, các mã nhân viên với nhiều kỹ năng được chọn. Đồng thời nếu một mã nhân viên chỉ có một hàng (mã 005), mã kỹ năng luôn luôn giống nhau nên nó không được chọn.

Câu hỏi con 3: [Đáp án đúng] a

Để chọn ra các tên nhân viên, bảng nhân viên là cần thiết và vì thế nó cũng là cần thiết để kết nối bảng nhân viên với bảng kỹ năng nhân viên. Cột cơ bản nhận được trong cả bảng nhân viên và bảng kỹ năng nhân viên là mã nhân viên. Sau đó, chúng ta kết nối mã nhân viên của bảng nhân viên (A) và bảng kỹ năng nhân viên (B1) sử dụng điều kiện AND. Bởi vậy cần thêm vào "`AND A.`MaNhanVien `= B1.`MaNhanVien"

Q3:

Điểm chính	➢ **Tổng khối lượng tính toán là hằng số nghĩa là khối lượng tính toán như nhau trong mọi trường hợp.** ➢ **Một danh sách liên kết 2 chiều có một con trỏ tiến và một con trỏ lùi.**

Một biến toàn cục là biến có thể tham chiếu đến từ mọi chương trình con. Một biến sử dụng trong một chương trình có thể là biến toàn cục hoặc biến cục bộ. Một biến cục bộ là biến chỉ hợp lệ trong phạm vi chương trình mà nó được khai báo.

Câu hỏi con 1: [Đáp án đúng] b, d

Yêu cầu "khối lượng (thứ tự) tính toán cần thiết" là "hằng số, bất chấp số dòng". Nó có nghĩa là khối lượng xử lý không thay đổi bất chấp thứ tự của các thủ tục và phép toán.

a) DELETE() xóa các dòng chỉ bởi CP. Bằng việc xóa, dòng kế tiếp và các dòng phía dưới sẽ đều được chuyển lên 1 dòng, bởi vậy lấy ví dụ xóa dòng Line 1 cần chuyển Line 2 → Line 1, Line 3 → Line 2, … , Line n → Line (n – 1), có (n-1) dịch chuyển. Nếu dòng Line 2 được chuyển, Line 1 ở nguyên vị trí cũ nhưng Line 3 → Line 2, Line 4 → Line 3, … , Line n → Line (n – 1), có (n-2) dịch chuyển. Bởi vậy, số dòng dịch chuyển phụ thuộc vào giá trị của n nên số lượng tính toán không phải hằng số.

b) GET() trả về xâu ký tự của dòng chỉ bởi CP. Nó chỉ đơn giản lấy ra xâu ký tự từ vị trí chỉ bởi CP nên số lượng tính toán là hằng số.

c) INSERT() thêm một dòng mới x vào dòng chỉ ra bởi CP. Bằng việc thêm vào, tất cả các dòng bên dưới dòng thêm vào dịch xuống dưới 1 dòng. Ví dụ, nếu 1 dòng được thêm vào trước dòng đầu tiên, Line n → Line (n + 1), Line (n – 1) → Line n, … , Line 1 → Line 2, có n dịch chuyển, và việc thêm và dòng Line 1 xảy ra. Nếu một dòng được thêm vào trước dòng thứ 2, Line 1 ở nguyên vị trí, nhưng Line n → Line (n + 1), Line (n – 1) → Line n, … , Line 2 → Line 3, có (n-1) dịch chuyển, và sau đó việc thêm vào Line 2 xảy ra. Bởi vậy, số lượng các dịch chuyển phụ thuộc vào giá trị của n nên khối lượng tính toán không là hằng số.

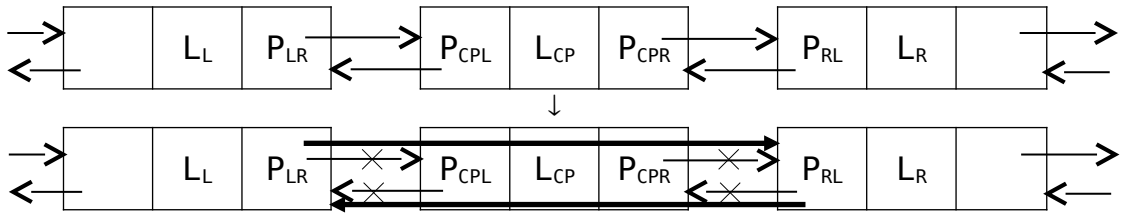
d) LAST() trả về số thứ tự n của dòng cuối (số lượng dòng của văn bản). Nếu văn bản rỗng, nó trả về 0. Như chỉ ra ở hình 1, nó được lưu ở TAIL, vì thế nó đơn giản lấy ra giá trị của TAIL. Bởi thế, khối lượng tính toán là hằng số.

Câu hỏi con 2: [Đáp án đúng] a, b, c

Cho một danh sách, việc xóa và thêm đơn giản là thay đổi con trỏ, vì thế cần thay đổi dòng là cần thiết. Trong các giải thích dưới đây, L_{CP} chỉ ra dòng cần xử lý (dòng chỉ bởi CP). Hơn nữa, L_L là dòng ngay trước nó và L_R là dòng ngay sau nó.

a) DELETE() thay đổi kết nối con trỏ như hình dưới đây. "x" chỉ ra một trỏ bị

xóa và một mũi tên đơn chỉ con trỏ được kết nối.



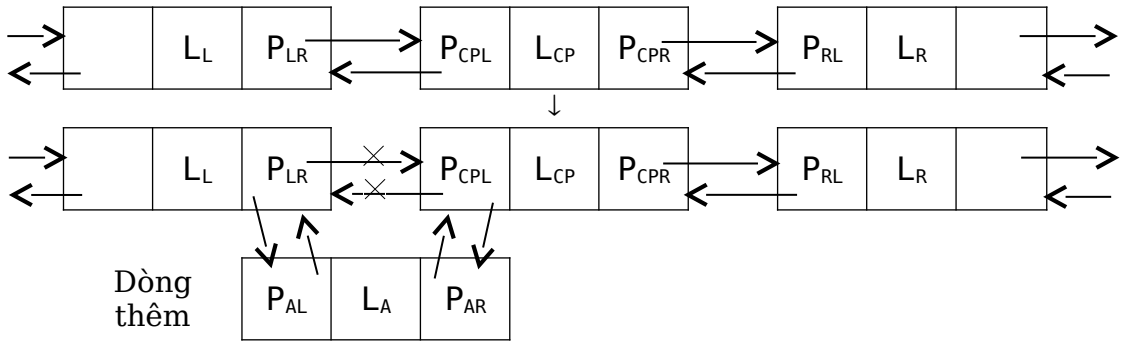
Việc thay đổi con trỏ được thực hiện như sau:

(1) P_{CPR} được chứa trong P_L sao cho L_L trỏ đến L_R vì L_{CP} bị xóa.
(2) P_{CPL} được chứa trong P_{RL} sao cho L_R trỏ đến L_L vì L_{CP} bị xóa.

Xử lý này chỉ bao gồm 2 phép toán (1) và (2) nên số lượng tính toán là hằng số.

b) GET() đơn giản đọc LCP tại vị trí được chỉ bởi CP, do đó số lượng tính toán là hằng số.

c) INSERT(x) thay đổi con trỏ kết nối như hình vẽ dưới đây. Giả sử rằng LA là dòng được thêm và con trỏ của dòng thêm vào đã biết.



Việc thay đổi con trỏ diễn ra như sau:

(1) P_{CPL} được chứa trong P_{AL} sao cho L_A trỏ đến L_L khi L_A được thêm.

(2) P_{LR} được chứa P_{AR} sao cho P_{AR} trỏ tới L_{CP} khi L_A được thêm.
(3) Giá trị con trỏ của dòng được thêm được chứa trong P_{CPL} sao cho L_{CP} trỏ tới L_A khi L_A được thêm vào
(4) Giá trị con trỏ của dòng được thêm được chứa trong P_{LR} sao cho L_L trỏ tới L_A khi L_A được thêm vào

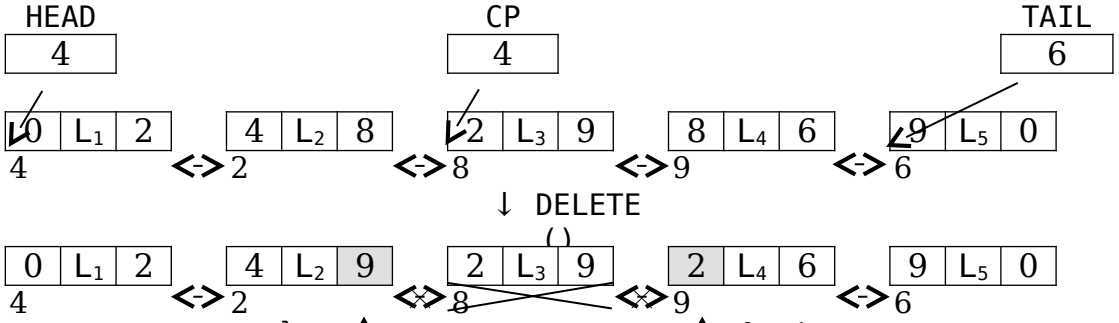
Quá trình này chỉ bao gồm các bước từ (1) đến (4) nên tổng khối lượng tính toán là hằng số.

d) Ở đây không có thông tin về số dòng ở đâu, do đó để tính tổng số dòng cần phải tiến hành tính toán thật, bắt đầu từ HEAD và các con trỏ tiếp theo. Hoặc, số dòng có thể tính từ TAIL. Do vậy, LAST() cần thời gian tính toán dài hơn khi có nhiều dòng hơn. Vì vậy khối lượng tính toán không

phải là hằng số.

Câu hỏi con 3: [Đáp án đúng] A – c, B – g, C – d

Hình vẽ dưới đây chỉ ra làm thế nào các dòng đi từ HEAD và TAIL. Việc xóa của L_3 dịch chuyển con trỏ như chỉ ra dưới đây:



Do đó “8 được chuyển thành 9” sao cho NEXT[2] trỏ đến L4, và “8 is changed to 2” sao cho PREV[9] trỏ đến L2.

Sau đó, chúng ta chuyển con trỏ của danh sách rỗng. Khi phần tử số 8 bị xóa, nó trở thành đầu của danh sách rỗng. Vì vậy EMPTY thay đổi “từ 3 thành 8”. Hơn nữa, chúng ta chắc chắn rằng NEXT[8] của phần tử số 8 đã xóa sẽ trỏ đến “3” (là đầu của danh sách rỗng trước khi xóa).

Q4:

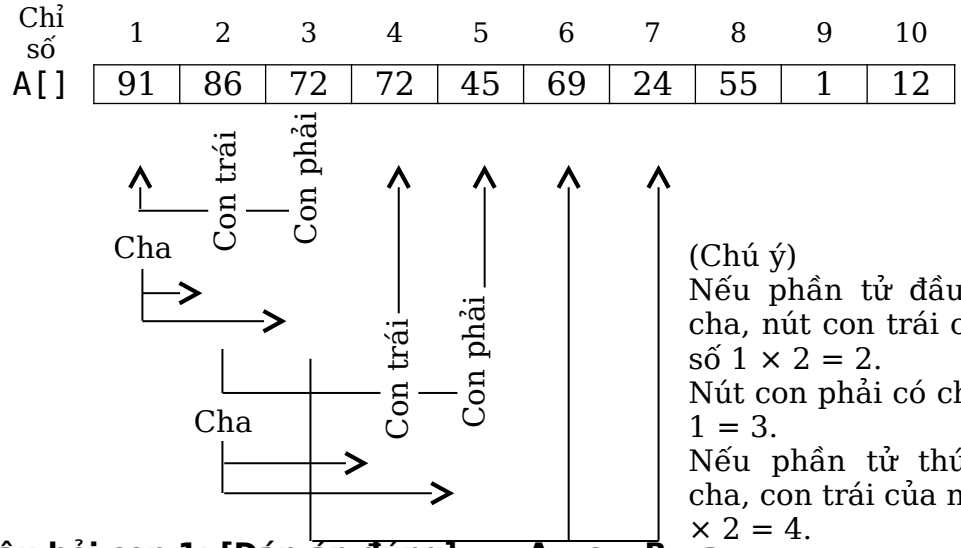
Điểm chính	➢ **Trong một đống, giá trị của gốc là lớn nhất và giá trị nút cha lớn hơn giá trị nút con.** ➢ **HeapSort lặp lại quá trình đưa nút gốc ra ngoài và vun đống cho cây còn lại.**

Đống là một cây nhị phân để lưu trữ dữ liệu từ nút nông đến nút sâu, ở cùng bậc của cây, từ trái qua phải, giá trị có những giới hạn sau:

Giá trị nút cha > giá trị nút con (hoặc giá trị nút cha < giá trị nút con)

Như vậy, các phần tử có giá trị lớn hơn (nhỏ hơn) được tập hợp xung quanh gốc và các phần tử có giá trị nhỏ hơn (lớn hơn) gần về phía các nút lá hơn. Vì gốc của cây chứa phần tử có giá trị lớn nhất (nhỏ nhất) nên cấu trúc dữ liệu này thích hợp cho việc xác định giá trị lớn nhất (nhỏ nhất).

Một đống có thể được biểu diễn bởi một mảng theo cách sau. Nếu một nút lưu trữ giá trị của phần tử `A[i]`thì nút con bên trái lưu trữ giá trị của phần tử `A[2×i]`, và nút con bên phải lưu trữ giá trị của phần tử `A[2×i + 1]`. Theo đó, mảng A chứa các phần tử như sau:



(Chú ý)
Nếu phần tử đầu tiên là nút cha, nút con trái của nó có chỉ số 1 × 2 = 2.
Nút con phải có chỉ số 1 × 2 + 1 = 3.
Nếu phần tử thứ hai là nút cha, con trái của nó có chỉ số 2 × 2 = 4.
Nút con phải có chỉ số 2 × 2 + 1 = 5.

Câu hỏi con 1: [Đáp án đúng] **A – c, B – a**

A:

Trong chương trình `HeapSort`, sau khi đổi chỗ giá trị nút đầu tiên và nút thứ `Idx`-th (`Swap(1, Idx)`) chương trình sẽ gọi chương trình con `Make Heap`. Chương trình con `Make Heap` vun cây còn lại thành đống. Để làm điều này, `Make Heap` so sánh các giá trị của nút cha, nút con trái, và nút con phải . Sau đó kết quả so sánh này được sử dụng để truyền tham số cho chương trình `Make Heap` khi gọi đệ qui.

Chỉ số của nút cha được lưu trữ trong biến `Top`, vì vậy các chỉ số của nút con trái và nút con phải cần phải được xác định trong ô trống "A" và câu lệnh tiếp theo. Nếu nút cha có chỉ số là `TOP` thì chỉ số của nút con trái là (`Top` × 2) và chỉ số của nút con phải là (`Top` × 2 + 1). Biểu thức "`R` ← `L` + 1" ngay sau ô trống "A" cho thấy `L` là chỉ số của nút con trái và `R` là chỉ số

của nút con phải. Vậy đáp án ở đây là biểu thức "L ← Top × 2" được chèn vào.

B:

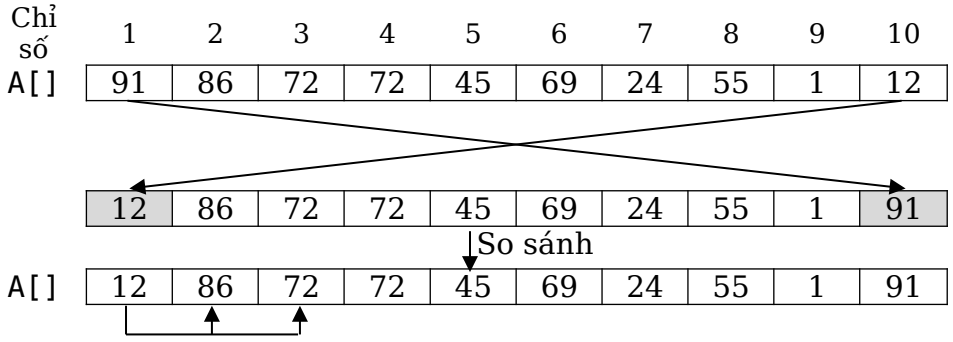
Biểu thức "R ≦ Last" không thể có trong trường hợp không có nút con phải bởi vì Last là chỉ số của phần tử cuối cùng và R là chỉ số của nút con phải. Trong trường hợp này, ta kiểm tra liệu có một nút con trái hay không. Nếu có, ta so sánh giá trị của nút cha (với chỉ số Top) và giá trị của nút con trái (với chỉ số L). Dù thế nào cũng có một nút con trái được xác định bởi "L ≦ Last" giống như "R ≦ Last" đã sử dụng ở phần trước trong chương trình. Do vậy, chúng ta chèn "L ≦ Last" ở đây.

Câu hỏi con 2: [Đáp án đúng] C – g, D – c

Theo thuật toán của chương trình con MakeHeap, ta sẽ tìm hiểu mảng trong biểu đồ đống được mô tả trong câu hỏi thay đổi như thế nào.

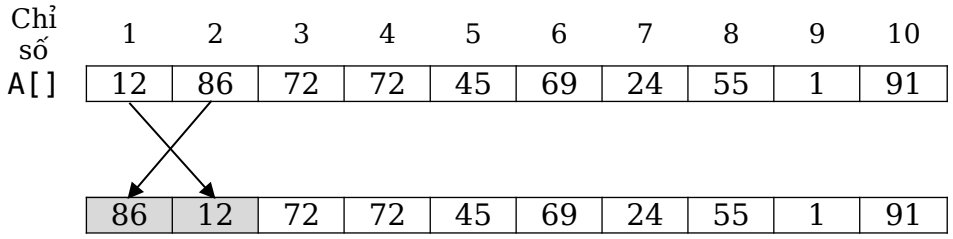
(1) Đổi chỗ lần thứ nhất

Gốc A[1] và phần tử cuối cùng A[10] được đổi chỗ. Sau đó, đống được vun lại.

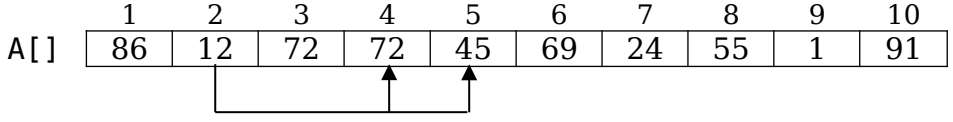


(2) Đổi chỗ lần thứ hai

Giá trị 86 (A[2], nút con trái) có giá trị lớn nhất theo kết quả so sánh, vì vậy A[1] và A[2]được đổi chỗ.

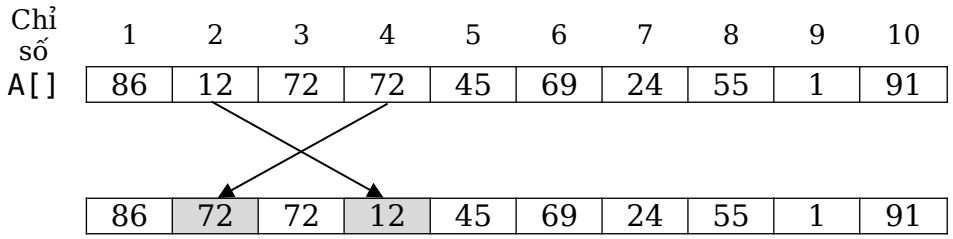


Tiếp theo, nút A[2] (nút vừa được đổi chỗ) coi là nút cha và quá trình xử lí tương tự xảy ra.

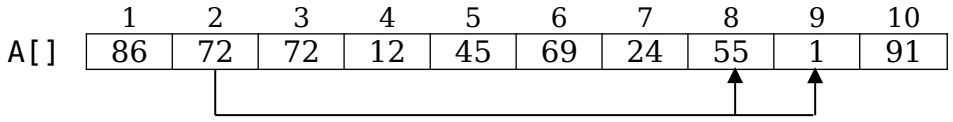


(3) Đổi chỗ lần thứ ba

Theo kết quả so sánh, 72 (A[4], nút con trái) có giá trị lớn nhất, vì vậy A[2] và A[4] được đổi chỗ.

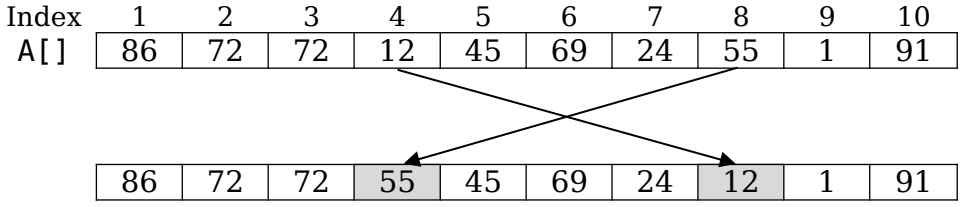


Tiếp theo, nút A[4] (nút vừa được đổi chỗ) coi là nút cha và quá trình xử lí tương tự xảy ra.



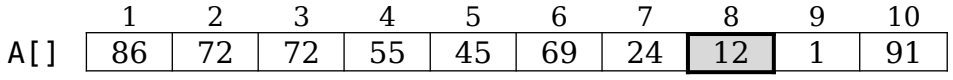
(4) Đổi chỗ lần thứ tư

Theo kết quả so sánh, 55 (A[8], nút con trái) có giá trị lớn nhất, vì vậy A[4] và A[8] được đổi chỗ.



Tiếp theo, coi A[8]là nút cha chúng ta tiếp tục so sánh, nhưng do A[8]không có nút con nào nên chương trình kết thúc ở đây. Vậy có tất cả 4 lần đổi chỗ giá trị các nút.

Trạng thái cuối cùng của mảng A được biểu diễn dưới đây, theo đó giá trị của 12 được lưu trữ trong phần tử mảng có chỉ số là “8.”



Theo trên thì có 4 lần đổi chỗ giá trị của các nút.

Câu hỏi con 2: [Đáp án đúng] C – g, D – c

Một đống được vun bằng việc tạo nút gốc (A[1]) có giá trị lớn nhất. Vì vậy ta sẽ vun đống cho cây con khi coi A[i] là nút cha. Ta tiếp tục vun đống cho các cây con trong khi giảm dần chỉ số đi 1 cho đến lúc vun tới nút A[1]; khi đó cây đã được vun thành đống. Một nút không có nút con nào thì không cần vun đống, nên giá trị khởi tạo của i là chỉ số của nút đầu tiên có một nút con.

Nút con trái của A[j] là (A[j × 2]), và nút con phải là (A[j × 2 + 1]), nên nút cha của nút của A[k] là A[k / 2]. Dẫn đến chỉ số của nút cuối cùng với một con là thương của số các phần tử (Last) cho 2, .., (Last / 2). Trong trường hợp cụ thể, khi Last = 10, chỉ số của nút cuối cùng có một nút con là (10 / 2 =) 5. Vậy ta cần vun lại đống mỗi lần chỉ số Idx giảm đi 1, với giá trị khởi tạo là (Last / 2), cho đến khi Idx = 1 (trong khi Idx ≧ 1). Do đó đáp án ở đây là “Idx: Last / 2, Idx >= 1, -1.”

Q5:

Điểm chính	➢ **Trong bước chuyển trạng thái, đích chuyển tiếp chỉ được xác định bằng trạng thái hiện tại và đầu vào.** ➢ **Chú ý rằng ban đầu cả cờ trắng và cờ đỏ đều hạ (down).**

Câu hỏi con 1: [Đáp án đúng] A – b, B – d, C – a

S0 là trạng thái bắt đầu, tại đó "cờ đỏ hạ (red flag down)" và "cờ trắng hạ (white flag down)." Điều này có thể kiểm tra bằng (2) của [Mô tả chương trình].

a) Theo đoạn mô tả của câu hỏi con, bước chuyển (i) là thao tác "nâng cờ đỏ (raise the red flag)" vì vậy trạng thái S1 là "cờ đỏ nâng (red flag up)" và "cờ trắng hạ (white flag down)."

b) Chuyển tiếp (ii) là từ S3 sang S0. Vì S0 là trạng thái "cờ đỏ nâng (red flag down)" và "cờ trắng hạ (white flag down)" nên S3 phải là một trạng thái có ít nhất một trong các cờ nâng (up). Mặt khác, S1 là trạng thái "cờ đỏ nâng (red flag up)" và "cờ trắng hạ (white flag down)" nên S3 phải là trạng thái "cờ trắng nâng (white flag up)" vả "cờ đỏ hạ (red flag down)."
Do đó bước chuyển (ii) là hành động "hạ cờ trắng (lower the white flag)."

c) S1 là trạng thái "cờ đỏ nâng (red flag up)" và "cờ trắng hạ (white flag down)," còn S3 là trạng thái "cờ đỏ hạ (red flag down)" và "cờ trắng nâng (white flag up)," vậy S2 là trạng thái "cờ đỏ nâng (red flag up)" và "cờ trắng nâng (white flag up)." Dưới đây là phần tổng kết của những trạng thái này.

Trạng thái (State)	Cờ đỏ (Red flag)	Cờ trắng (White flag)
S0	Hạ (Down)	Hạ (Down)
S1	Nâng (Up)	Hạ (Down)
S2	Nâng (Up)	Nâng (Up)
S3	Hạ (Down)	Nâng (Up)

Do đó, bước chuyển (iii) là " nâng cờ đỏ (raise the red flag)."

Câu hỏi con 2: [Đáp án đúng] a, g

Thông tin trong tệp giá trị ban đầu, theo (2) của [Giải thích chương trình] bao gồm (thời gian phát hiện câu trả lời, thời gian hiển thị cửa sổ cho trả lời sai, số lần lặp lại).

Quá trình sử dụng thời gian tìm ra câu trả lời có thể được xác định như sau. Chú ý (4) của [Giải thích chương trình]: *Có 5 loại thao tác có thể nhận được từ các thiết bị nhập:* nâng cờ đỏ, hạ cờ đỏ, nâng cờ trắng, hạ cờ trắng, và không di chuyển cờ nào (khi không phát hiện ra câu trả lời nào trong thời gian phát hiện câu trả lời).

Mô tả trên giúp ta thấy rõ rằng thời gian phát hiện câu trả lời là cần thiết

trong quá trình liên quan đến các thao tác này. Quá trình này là quá trình "phát hiện trả lời".

Bên cạnh đó, quá trình sử dụng thời gian hiển thị cửa sổ cho trả lời sai có thể được xác định như sau. Chú ý (5) của phần [Giải thích chương trình]:

Chương trình sẽ phát hiện ra câu trả lời của người chơi. Nếu đúng, chương trình sẽ cộng thêm "1" vào số câu trả lời đúng và gửi cửa sổ câu trả lời đúng tới thiết bị xuất. Nếu sai, chương trình sẽ gửi cửa sổ câu trả lời sai tới các thiết bị xuất và sau khi hết thời gian hiển thị cửa sổ câu trả lời sai trong tệp giá trị khởi tạo, chương trình sẽ thiết lập cờ trong cửa sổ này sang trạng thái đúng.

Mô tả trên giúp ta thấy rõ rằng thời gian hiển thị trả lời sai là cần thiết trong quá trình liên quan đến các thao tác này. Quá trình này là quá trình "xử lý trả lời sai."

Q6:

Điểm chính	➢ **Ký tự đầu tiên là "<," nên phải duyệt từ ký tự thứ hai.** ➢ **Trong vòng lặp "for" chỉ số sẽ dừng khi gặp ký tự ">," nên phải duyệt từ ký tự tiếp theo.**

Câu hỏi con: [Đáp án đúng] A – a, B – c, C – a, D – d

Chương trình tuy phức tạp nhưng với ví dụ cụ thể trong Hình 2 và Bảng 2 thì đoạn chương trình trở nên dễ hiểu hơn. Thêm nữa, những chú thích trong chương trình là phù hợp. Theo [Mô tả chương trình] (2) (iii) chỉ rõ rằng không có lỗi cú pháp nào trong các xâu ký tự gắn thẻ, vì vậy một xâu ký tự bắt đầu với ký tự "<" có nghĩa bắt đầu một thẻ trong khi ký tự ">" thì xác định kết thúc thẻ, và hai ký tự đó luôn tạo thành một cặp. Trong chương trình, phần "XXXX" trong "<XXXX>" được lưu trữ như tên của thẻ. Như vậy, liên hệ (một cách tương ứng) câu lệnh "`mlstr = parse_ml_date(mlstr +1, 1);`" với câu lệnh "`char *parse_ml_date(char *mlstr, int level)`", ta thấy quá trình duyệt bắt đầu từ ký tự thứ hai của `mlstr` và độ sâu bắt đầu từ 1. Ở đây giả thiết rằng ký tự đầu tiên của `mlstr` là "<".

A:
Tham khảo Bảng 2, con trỏ trỏ tới "`STUDENT`" được lưu trữ trong `elmtbl[elmnum].tag`. Giá trị khởi tạo của `elmnum` là 0, vì vậy con trỏ trỏ tới "`STUDENT`" (cũng là địa chỉ của ký tự "S") được lưu trữ trong `elmtbl[0].tag`. Vậy đáp án đúng chèn vào ô trống là "`mlstr`".
Hơn nữa, trong câu lệnh tiếp theo sẽ lưu trữ chiều sâu của thẻ và con trỏ của xâu duyệt qua các ký tự cho đến khi gặp ký tự ">." Trên thực tế ký tự "\0" đươc lưu trữ ở vị trí của ">" chỉ định rằng quá trình lưu trữ tên thẻ đã hoàn thành.

B:
Đây là quá trình lưu trữ giá trị thẻ, vì vậy theo Bảng 2, ta thấy con trỏ trỏ tới

"BILL" được lưu trữ trong elmtbl[elmnum].value. Hơn nữa, trong vòng lặp "for", con trỏ xâu dừng tại vị trí của ">." Do đó khi tăng giá trị của con trỏ 1 thì sẽ trỏ tới vị trí của ký tự "B". Vậy đáp án đúng sẽ là "mlstr + 1".

C:
Sau khi lưu trữ giá trị thẻ, ta duyệt tiếp đến ký tự "<" hoặc "</." Điều này thể hiện trong vòng lặp "for" sau ô trống B. Ngay khi bắt đầu phân tách thẻ, con trỏ dừng tại vị trí của ký tự "<," vì vậy chương trình cần xác định ký tự tiếp là kí tự "/" hay không. Nếu ký tự tiếp là "/" thì chiều sâu thẻ không thay đổi, ngược lại thì tồn tại thẻ khác mức thấp hơn ở bên trong. Vậy ở đây ta cần tăng chỉ số mảng lưu trữ lên 1 và đáp án đúng chèn vào ô trống là "elmnum++".

D:
Theo giải thích ở đáp án C, nếu ký tự tiếp theo không phải là "/" thì tồn tại thẻ khác mức thấp hơn ở bên trong. Trong trường hợp này, hàm "parse_ml_date" được gọi đệ qui, và sự phân tích xâu lại tiếp tục. Vậy ở đây, ta chèn vào ô trống điều kiện ký tự tiếp theo không phải là "/."
và đáp án là "*(mlstr + 1) != '/'".

Q7:

Điểm chính	➢ **Luồng (thread) được bắt đầu như một đơn vị thực thi riêng rẽ và xảy ra đồng thời với phương thức main.** ➢ **Chiều dương cho x là bên trái, chiều dương cho y là xuống dưới.**

Câu hỏi con 1 : [Đáp án đúng] A – c, B – d

Một luồng là một đơn vị của quá trình thực thi chương trình, và nó có thể được bắt đầu như một đơn vị thực thi riêng rẽ xảy ra đồng thời với phương thức main (phương thức main được thực thi ngay từ đầu). Trong Java, có 2 cách để tạo tạo một luồng. Cách thứ nhất là thừa kế lớp Thread, và cách thứ hai là cài đặt giao diện Runnable. Chương trình này sử dụng cách thứ hai.

A:
Trong lớp trừu tượng Motion, thêm vào cho phương thức khởi tạo Motion và phương thức "run()", được sử dụng để tiếp tục hiển thị chuyển động của điểm, phương thức trừu tượng "update" được định nghĩa để xác định chuyển động của điểm. Trong phương thức "run()", một điểm được vẽ tại vị trí tọa độ được định nghĩa bởi "while(true){…}". Câu lệnh "while" thực hiện việc sau: (1) vẽ điểm, (2) dừng 40 mili giây, (3) ô trống A, và (4) xóa điểm. Ở đây, có nhu cầu di chuyển điểm giữa (2) và (4) ở trên, vì vậy trong ô trống A cần gọi phương thức "update()" để di chuyển điểm và thay đổi nội dung của điểm. Do đó, chúng ta cần chèn "point = update(point)" Chú ý rằng "Point current = point;" ngay trước ô trống A là quá trình rút lại để xóa điểm trước khi di chuyển với "Space.erase(current)" ngay sau ô trống A.

B:
Ô trống B nằm bên trong "new Thread (…)" của "for (int i = 0; i <

point.length; i++)" là một quá trình được lặp lại. Điều này chỉ ra rằng số luồng sẽ được tạo ra tương ứng với số điểm được vẽ. Với một luồng, như đã được đề cập ở trên thì giao diện Runnable phải được cài đặt, nhưng lớp Motion là một lớp trừu tượng và không thể được sinh ra như một luồng. Nói cách khác, lớp SimpleMotion, thừa kế lớp trừu tượng Motion, được sinh ra như một luồng. Do đó "new SimpleMotion(points[i])" được chèn ở đây.

Câu hỏi con 2 : [Đáp án đúng] b

Trong hình vẽ của câu hỏi con 2, ở góc trái dưới cùng được đặt là gốc tọa độ (0, 0). Do đó, các dấu hiệu ban đầu cho mỗi chuyển dời (i), (ii), (iii) và (iv) như sau:

(i) X âm, Y dương
(ii) X dương, Y dương
(iii) X âm, Y âm
(iv) X dương, Y âm

Từ [Chương trình 3], ta thấy trong phương thức "update()", các giá trị x và y được tính toán từ directionX và directionY tương ứng. Dòng 2 định rõ các giá trị ban đầu của directionX và directionY đều là 1. Vì vậy X và Y đều dương. Vì thế cả x và y đang di chuyển theo chiều tăng, tức là mũi tên (ii). Câu trả lời là "Điểm P di chuyển theo chiều mũi tên (ii)."

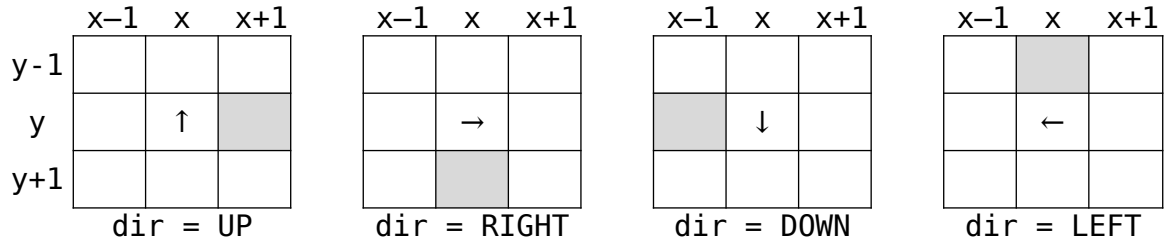
Q8:

Điểm	➢ **Các hướng chuyển động là ↑ đi lên, → sang phải, ↓ đi xuống, và ← sang trái.** ➢ **"rcheck( )" là hàm dùng để xác minh xem nó có thể chuyển động sang bên phải không.**

Chú ý đầu tiên đó là trong các dòng từ một đến 4, UP, RIGHT, DOWN, và LEFT được định nghĩa là các hắng số biểu diễn các hướng đi lên, sang phải, đi xuống và sang trái và tương ứng với các giá trị là 0, 1, 2, và 3. Câu hỏi bản thân nó không cho bất cứ chỉ định về các lời hướng dẫn nhưng tên của chúng lại cho các manh mối. Trong câu hỏi 1, điều kiện khởi tạo là x = 1 và y = 0, dấu chấm "Điểm bắt đầu" của hình 1. Ở đây, hướng chuyển động có thể chỉ là đi xuống, vì vậy bạn nhìn thấy rằng DOWN thực sự nghĩa là hướng xuống.
Bây giờ, quan tâm đến các xử lý của các hàm "rcheck() ", "fcheck()", và "go()".

(1) rcheck( )

Khi "dir = UP" hàm này sẽ trả về "M[y][x + 1]", vì vậy nó đang trả về tạo độ của hình vuông bên phải. Nếu "dir = RIGHT" hàm này sẽ trả về "M[y + 1][x]", vi vậy nó đang trả về tạo độ của hình vuông bên dưới. Nếu "dir = DOWN" hàm này sẽ trả về "M[y][x − 1]" vi vậy nó đang trả về tạo độ của hình vuông bên dưới, và khi "dir = LEFT" hàm này sẽ trả về "M[y − 1][x]", đây là tọa độ hình vuông bên trên nó. Trong các hình vẽ bên dưới, mỗi mũi tên (↑, →, ↓, và ←) chỉ định các hướng được xác minh bởi giá trị của "dir," và hình vuông tối là vị trí của các tọa độ trả về.



Trong mỗi hình vẽ ở trên, nếu mỗi vùng xám hoặc là ROAD hoặc là EXIT, thì chương trình có thể xử lý hình vuông tối. Vì vậy để xử lý hình vuông tối, chúng ta phải đặt "dir = (dir + 1) %4". Bảng bên dưới chỉ định những biến đổi của "dir" cho hoạt động này.

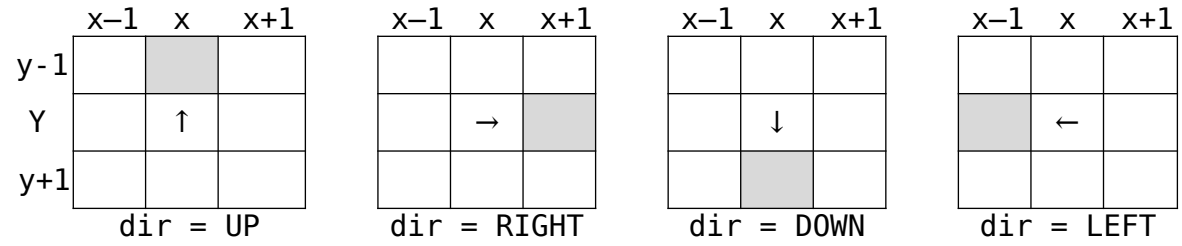
"dir" hiện tại	Gía trị	(dir + 1) %4	"dir" mới
↑	0	1	→
→	1	2	↓
↓	2	3	←
←	3	0	↑

Các điểm "dir" mới đến hình vuông bên phải của hướng chuyển động. Kể từ đây, chương trình đầu tiên sẽ kiểm tra xem nó có thể chuyển đến hình vuông bên phải không. Chú ý rằng nếu nó có thể, điểm sẽ chuyển đến hình vuông đó, nếu không chương trình gọi hàm fcheck() và quyết định bên trong hàm đó hướng của điểm nên di chuyển tiếp theo.

(2) fcheck()

Quan tâm điều chỉnh trong hàm rcheck().

```
dir = UP:       M[y − 1][x]     → Trả về tọa độ hình vuông bên trên
dir = RIGHT:    M[y][x + 1]     → Trả về tọa độ hình vuông bên phải
dir = DOWN:     M[y + 1][x]     → Trả về tọa độ hình vuông bên dưới
dir = LEFT:     M[y ][x − 1]    → Trả về tọa độ hình vuông bên trái
```



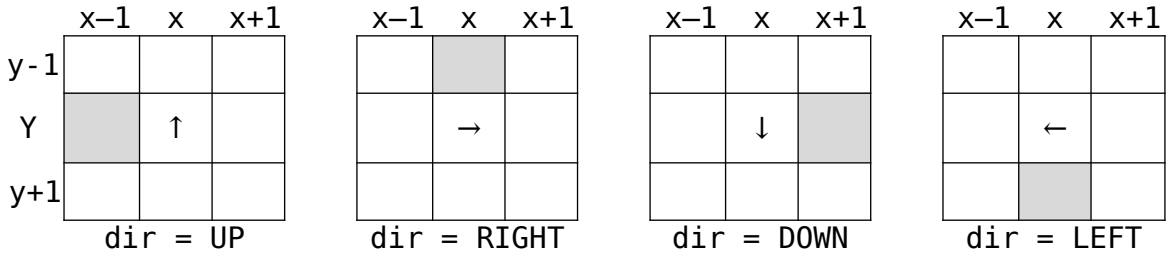
Nếu vùng tối trong mỗi hình ở trên hoặc là ROAD hoặc là EXIT, thì chương trình có thể hướng vào vùng tối, vì vậy chương trình gọi trực tiếp hàm "go()" và tiếp tục. Điều này đang kiểm tra có hay không có thể đi đến hình vuông tiếp theo trong hướng chuyển động.

Kể từ đây, bạn có thể nhận thấy rằng nếu hàm rcheck() xác minh được rằng hướng đến hình vuông bên phải là không thể thì chương trình sẽ kiểm tra xem nó có thể chuyển đến hình vuông tiếp theo trong hướng của chuyển động.

Bây giờ, nếu chương trình gọi hàm rcheck() hoặc fcheck() và tìm thấy rằng không có chyển động nào có thể thì bạn có thể nhìn thấy từ dòng 28, giá trị của "dir" được thay đổi trong theo hướng của hình vuông bên trái. Khi điều này xảy ra, hàm "go()" không được gọi, ví nó chỉ lien quan đến thay đổi của "dir."

"dir" hiện tại	Gía trị	(dir + 3) %4	"dir" mới
↑	0	3	←
→	1	0	↑
↓	2	1	→
←	3	2	↓

Mối quan hệ giữa hướng hiện tại và hướng mới (tối) theo sau. Chú ý rằng hướng được trả về bằng góc 90 độ bên trái (Ngược chiều kim đồng hồ).



(3) go()

Mối quan hệ giữa giá trị của "dir" và hướng chuyển động theo sau. Chú ý rằng, không giống "rcheck()", "fcheck()" và "go()" thay đổi thực sự các tọa độ.

```
dir = UP     :y--          → Chuyển đến hình vuông bên trên
dir = RIGHT  :x++          → Chuyển đến hình vuông bên phải
dir = DOWN   :y++          → Chuyển đến hình vuông bên dưới
dir = LEFT   :M[y][x – 1]  → Chuyển đến hình vuông bên trái
```

Câu hỏi 1 : [Trả lời dúng] a

Trước tiên, chương trình gọi hàm "rcheck( )" và xem nó có thể đi tiếp không. Bởi vì giá trị khởi tạo của "dir" là DOWN (= 2), chương trình kiểm tra hình vuông bên phải của hướng chuyển động (đó là M[0][0]). Đi vào theo hướng đó là không thể, vì vậy chương trình gọi hàm "fcheck()" để kiểm tra hướng chuyển động (hình vuông bên dưới, đó là hình vuông 1). Bởi vì nó có thể chuyển đến hình vuông 1, hàm "go()" được gọi, và vị trí chuyển đến M[1][1] (hình vuông 1).

Tiếp theo, hàm "rcheck()" được gọi lại, và hình vuông bên phải của hướng chuyển động (M[1][0]) được kiểm tra, nhưng lại một lần nữa hình vuông đang chuyển động không được cho phép. Vì vậy hàm "fcheck()" được gọi, và rồi hình vuông trong hướng chuyển động M[2][1]) được kiểm tra. Tiếp tục vì hình vuông đó là có thể, vì vậy bây giờ chương trình chuyển đến M[2][1]. Những xử lý này được lập lại, và đường chuyển động tử hình vuông 1 đến hình vuông 2.

M [0] [1]

	↓						
	1		3	4		8	
	↓						
	↓						
	2	b	5		7		
	a						
	6						

Tại M[4][1], chương trình không thể đi đến bất cứ nơi đâu bởi hàm "rcheck()" hoặc "fcheck()", vì vậy câu lệnh trong dòng 28 được thực thi. Hàm "go()" trả về hướng chuyển động bằng góc 90 độ bên trái, vì vậy hướng được thay đổi đến dir = 1 (RIGHT, →). Tiếp đến chương trình gọi hàm "rcheck()" và hình vuông bên phải của hướng chuyển động (hình vuông bên dưới, M[5][1], giữa hình vuông 2 và hình vuông 6) được kiểm tra. Tuy nhiên nó không thể chuyển đến đó, vì vậy hàm "fcheck()" được gọi để kiểm tra hình vuông trong hướng chyển động (M[4][2], giữ hình 2 và hình 5). Đây cũng là hình vuông không thể chuyển động đến. Lại một lần nữa, dòng lệnh 28 được thực thi, trả về hướng chuyển đôngj bởi góc 90 độ bên trái, thành hướng lên "↑." Trong trạng thái này, hướng lên "↑" sẽ chuyển điểm từ hình 2 đến hình một. Tuy nhiên tại hình 1, chương trình kiểm tra hình vuông bên phải của hướng ↑ của chuyển động, nguyên nhân nó chuyển hướng đến hình vuông 3. Chình tự, chúng ta có ở bên dưới:

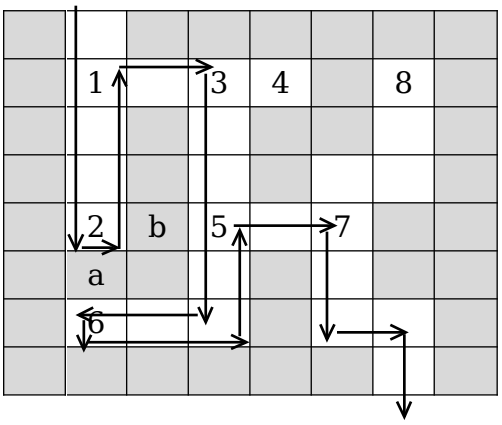
Điểm bắt đầu → hình vuông 1 → hình vuông 2 (Rồi trả về)
hình vuông 2 → hình vuông 1 (Rồi trả sang phải)
hình vuông 1 → hình vuông 3 (Tại hình vuông 3, hướng là "→", vì vậy nó chuyển sang bên phải, đó là hướng xuống).
hình vuông 3 → hình vuông 5 → ("↓") → M[5][3] → hình vuông 6
(Tại hình vuông 5, nó chuyển vào hướng "↓").
hình vuông 6 → ("→") → M[6][3]→ hình vuông 5
(Tại hình vuông 6, một trả về 90 độ bên trái

xảy ra hai lần, thay đổi hướng "→." Tại `M[6][3]`, nó trả về hướng "↑" .)

hình vuông 5 → hình vuông 7 (Tại hình vuông 5, hướng thay đổi đến bên phải, hướng đến hình vuông 7)

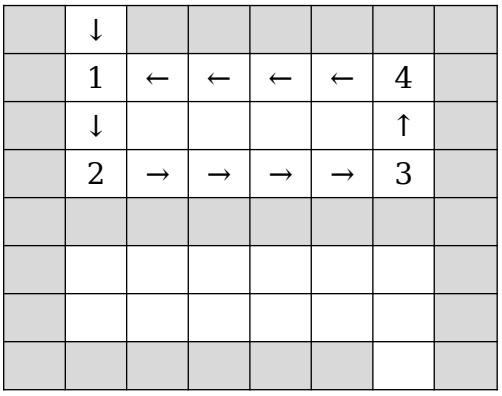
hình vuông 7 → điểm cuối.

Vì vậy, đường đi là "Điểm đầu→*1*→*2*→*1*→*3*→*5*→*6*→*5*→*7*→điểm cuối."



Câu hỏi 2 : [Trả lời đúng] b

Như diễn giải trong câu hỏi 1, chúng ta chỉ định sự di chuyển. Ở đây nó di chuyển như thế nào:



Hình vuông 1: Chương trình mong đợi bên phải của hướng di chuyển, nhưng nó không thể di chuyển sang bên phải. Vi vậy nó hướng vào trong hướng của chuyển động.

Hình vuông 2: Chương trình mông đội sang bên phải, nhưng nó không thể chuyển đến đó. Vì vậy nó mông đợi hướng tới, nhưng chuyển động hướng tới cũng không thể. Ví vậy nó thay đổi hướng sang trái (→).

Hình vuông 3: Lại một lần nữa, chương trình kiểm tra hình vuông bên phải và hình vuông đầu đằng trước. Không có chuyển động nào lá có thể. Vì vậy nó thay đổi sang hướng bên trái của hướng chuyển động, thay đổi ↑.

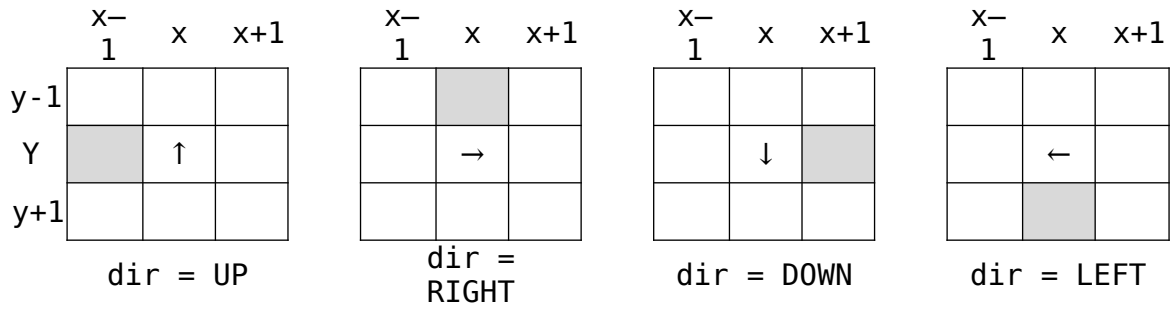
Hình vuông 4: Lại một lần nữa, chương trình kiểm tra hình vuông bên phải và hình vuông đầu đằng trước. Không có chuyển động nào lá có thể. Vì vậy nó thay đổi sang hướng bên trái của hướng chuyển động, thay đổi ←.

Vì vậy, chuyển động *1→2→3→4* được lập lại. Câu trả lời rồi là "Điểm bắt đầu→*1→2→3→4→1→2→3→4*→... được lập lại vô tận."

Câu hỏi 3 : [Trả lời đúng] a

Sau câu lệnh "`while`" (dòng 20), chúng ta them một câu lệnh "`print`". Bây giờ, từ điều kiện mang vào trong câu lệnh "`while`" là "`M[y][x]! = EXIT`", chương trình sẽ thoát câu lệnh điều kiện "`while`" khi "`M[y][x] = EXIT`". Rồi, từ điểm cuối "`M[y][x]`" không được in, một câu lệnh "`print`" cũng được yêu cầu thêm vào sau dòng 29.
Vì vậy, một câu lệnh "`print`" nên được thêm trực tiếp sau mỗi dòng 20 và 29.

Câu hỏi 4 : [Trả lời đúng] b
Xử lsy của hàm "`lcheck()`" như sau:



Trái ngược với hàm "`rcheck()`" kiểm tra phía bên phải của hướng chuyển động, hàm "`lcheck()`" kiểm tra phía bên trái. Chương trình dung hàm "`lcheck()`" đê xem nó có thể chuyển sang bên trái không. Nếu nó có thể, hướng của chuyển động được trả về bằng góc 90 độ bên trái. Vì điều này, chúng ta thay đổi dòng 23 để mà điều chỉnh giống dòng 28, chuyển động lấy trả về bởi góc 90 độ bên trái. Từ khi dòng 23 được thay đổi để làm trả về góc 90 độ bên trái, dòng 28 cần được thay đổi để mà quay hướng bên phải bằng góc 90 độ.
Kể từ đây, chúng ta nhận thấy rằng những thay đổi sau là cần thiết:

(Số dòng)
23 `dir = (dir+1) % 4;` → `dir = (dir+3) %4;` /* 90 độ bên trái */
28 `dir = (dir+3) % 4;` → `dir = (dir+1) %4;` /* 90 độ bên phải */

Vì vây, các thay đổi được chỉ định: "+1 trong dòng 23 phải được thay đổi thành +3 và +3 trong dòng 28 phải được thay đổi thành +1."

Q9:

Điểm	➢ Loại sự kiện là DIGIT, OPERATOR, và CLEAR. ➢ CLEAR có một ít các tham số.

Vai trò của mỗi lớp như sau:

(1) Lớp `CalculatorEvent`
Đây là lớp khi một nút tính toán được nhấn. Có hai hàm khởi tạo gọi là "`CalculatorEvent`" có điều này là bởi vì có trường hợp một tham số truyền vào là cần thiết và có trường hợp tham số truyền vào là không cần thiết, phụ thuộc vào loại phím được nhấn.

(2) Giao diện `CalculatorOutput`
Đây là giao diện chỉ để khai báo hai phương thức trừu tượng "`display`" với các loại tham số khác nhau, được dùng như là các tham số của hàm khởi tạo của lớp `Calculator`.

(3) Lớp `Calculator`
Đây là lớp tính toán bản thân nó. Chú ý rằng tham số khởi tạo được định nghĩa là một biến của loại giao diện `CalculatorOutput`. Khi lớp này lấy đối tượng cụ thể, biến loại giao diện được chỉ định sẽ lấy trực tiếp thay thế bên trong đầu ra biến loại giao diện.

(4) Lớp `CalculatorTest`
Đây là lớp để kiểm tra lớp `Calculator` và ở đây phương thức "`main`" được định nghĩa. Chú ý rằng tham số cần thiết để xây dựng một đối tượng lớp `Calculator` được miêu tả, bao gồm ô trống B.

Câu hỏi 1 : [Trả lời chính xác] A – e, B – c, C – e

A:
Ô trống A lá xử lý khởi tạo khi chỉ có một tham số (`int type`). Khi "`type`" is `CLEAR` (phím clear), tham số thứ hai "`value`" không được dùng, vì một vài giá trị phù hợp đã được đặt vào cho "`value`" của tham số thứ hai, và những tham số này gọi hai hàm khởi tạo. Cho tất cả lựa chọn trong nhóm câu trả lời, tham số hai là 0. Cũng vậy, cho các tham số gọi hai hàm khởi tạo, chúng sử dụng từ khóa "`this`" để tham chiếu đến bản thân chúng. Từ đây, "`this(type,0)`" được thêm vào.

B:
Ô trống B là tham số khi đối tướng "`calc`" của lớp `Calculator` được tạo. Dựa trên việc `CalculatorOutput` được đóng gói là lớp nặc danh nào và do đó phương thức "`display`" đầu ra là giá trị số được chỉ định hoặc xâu kí tự đến `System.out` trong gói này, một chỉ định có thể đưa ra đó là ô trống B phải tạo một lớp nặc danh được đóng gói giao diện `CalculatorOutput`. Ngoài ra, Tham số được truyền đến hàm khởi tạo của lớp `Calculator` là một biến của loại giao diện và lớp đóng gói giao diện "`CalculatorOutput`" phải đóng gói hai loại phương thức (một cho mỗi loại) "`display`."

Vì những lý do này, trong phần chứa đụng ô trống B này, giao diện `CalculatorOutput` được đóng gói là một lớp không đặt tên (lớp nặc danh), và, tại cùng thời điểm, nó phải được cụ thể bằng một từ khóa mới. Từ đay, câu trả lời cần bao gồm tên của giao diện được đóng gói bởi một lớp nặc danh. Câu trả lời là "`new CalculatorOutput()`."

Tình cờ, một lớp nặc danh được dùng trong các tình huống đa dạng, như là khi chúng ta lấy một lớp tồn tại hoặc giao diện, thay đổi nó một phần để tạo một lớp mới, và ước mong dùng lớp mới này cục bộ trong một lớp. Như trong một trường hợp, gói và kế thừa lấy đặt bên ngoài các từ khóa "`extends`" hoặc "`implements`".

C:
Ô trống C phù hợp với các tham số được dùng khi trường hợp "`event`" được tạo khi các giá trị đầu vào là 0 đến 9. Tham số được truyền vào, quan tâm đến các trường hợp giống như = và +, và tham số đầu tiên là `CalculatorEvent.DIGIT`. Cho tham số thứ hai, chương trình có thể đơn giản truyền vào số được nhập, nhưng các kí tự từ "0" đến "9" cần được chuyển đổi đến các số từ 0 đến 9. Khi các kí tự "0" đến "9" phù hợp với 0x30 với 0x39 trong số hexa, chuyển đổi đến các giá trị số cs thể hoàn thành đơn giản bằng việc trừ đi số hexa 0x30. Vì vậy, những gì cần được thêm vào ô trống C là "`CalculatorEvent.DIGIT, c - '0'`."

Câu hỏi 2 : [Trả lời đúng] D – a, E – d, F – i

D:
Đây là trường hợp ở đây đầu vào là những kí tự xâu "`3*4***=`". Khi từ khóa xử lý được vào nhiều lần liên tiếp, câu lệnh tiếp theo sau khi hiển thị kết quả hoạt động thực thi "`register = 0;`", thiết lập biến để nhớ kết quả ("`register`") đến 0. Kể từ đây, thậm chí nếu từ khóa bằng được nhận, kết quả hiển thị là 0.

E:
Tại thời điểm "`3*4 =`" được nhấn, đồng thời hiển thị và giá trị của biến "`accumulator`" là 12. Khi "`+5`" chỉ định, Gía trị hiển thị ở cuối là giá trị vào cuối "5." Đấy là bởi vì "`register`" được gán bằng 0 khi kí hiệu bằng được nhấn. Với "register" 0, cho đến khi số khác được thêm đến đầu vào giá trị 5, hiển thị chỉ định "5." Từ đây, kết quả hiển thị là "5."

F:
Tại thời điểm "`3 + 4 / 0`" được vào, "`accumulator`" chưas 7, đó là kết quả của tính toán "`3 + 4`", trong khi "`register`" giá trị đầu vào cuối 0. Rồi, khi từ khóa bằng được nhấn, chia cho 0 xảy ra, kích hoạt xử lý ngoại lệ "`ArithmeticException`". Từ đây, "`Error`" được hiển thị.